P. 1877/72

3 1972

informatyka

SPIS TREŚCI

Str.

Thanasis Kamburelis — „System ODRA 1305 — Organizacja logiczna i podstawowe dane użytkowe" 1

Marek Hołyński — „Przewidywanie tendencji zmian procesów za pomocą łańcuchów Markowa" 9

Ryszard Frydrychowski — „Programowanie liniowe w liczbach całkowitych" 13

Piotr Perkowski — „ZAM-GPSS — język do symulacji procesów dyskretnych" 16

Thomas E. Cheatham — „Dotychczasowy rozwój języków programowania" — tłum. z ang. Jowita Koncewicz 20

TRYBUNA CZYTELNIKA

List Stefana Zawadzkiego w sprawie terminologii 24

Z KRAJOWEGO BIURA INFORMATYKI

Komu komputer? 28
Przegląd prasy krajowej 28
Współpraca z zagranicą 29
Przegląd prasy zagranicznej 31

ZE ŚWIATA

Brytyjskie Towarzystwo Komputerowe — opracował A. B. Empacher . . 32
Informatyka w bułgarskim przemyśle maszynowym — D.P. . . . . III okł.

PRZEGLĄD WYDAWNICTW

Infrainformator ISIS — ABE. . . . . IV okł.
Bibliografia książek polskich z dziedziny informatyki . . . . . IV okł.

WYDAWNICTWA CZASOPISM TECHNICZNYCH NOT
Warszawa
Czackiego 3/5

KOLEGIUM REDAKCYJNE
Redaktor naczelny prof. dr Leon ŁUKASZEWICZ

Doc. dr hab. inż. Konrad FIAŁKOWSKI (zast. redaktora naczelnego), Władysław KLEPACZ, dr Antoni MAZURKIEWICZ, inż. Dorota PRAWDZIC (zast. redaktora naczelnego), dr inż. Andrzej TARGOWSKI

Sekretarz Redakcji mgr Krystyna Wrońska

RADA PROGRAMOWA

Mgr inż. Jan Bursche, mgr inż. Henryk Chyrek, (wiceprzewodniczący) mgr inż. Ryszard Dąbrówka, mgr inż. Bolesław Gliksman, mgr inż. Józef Knysz, prof. dr Leon Łukaszewicz, mgr inż. Jan Matejak, prof. dr Tadeusz Peche (przewodniczący), mgr inż. Jerzy Trybulski (wiceprzewodniczący), dr Tadeusz Walczak, mgr Kazimierz Wasilewski, mgr Waldemar Wiśniewski (sekretarz), mgr Stefan Wojciechowski, dr inż. Henryk Woźniacki, mgr inż. Jan Zdzisław Żydowo

Redakcja: Warszawa, ul. Jasna 14/16, pokój 332, tel. 26-82-61, w. 285. Zastępca redaktora naczelnego tel. 28-37-29

Zakład Kolportażu WCT NOT, Warszawa, ul. Mazowiecka 12

Zakł. Graf. „Tamka". Z. 2. Zam. 51. Papier druk. sat. IV kl. 70 g, 61 × 86. Obj. 4 ark. druk. Nakład 3350. A-103.

Cena egzemplarza zł 8.— **INDEKS 36707** **Prenumerata roczna zł 96.—**

P.1877/72

Informatyka

dawniej Maszyny Matematyczne

zastosowania w gospodarce, technice i nauce

Nr 3

MIESIĘCZNIK

1 9 7 2

ROK VIII

Marzec

ORGAN KRAJOWEGO BIURA INFORMATYKI I POLSKIEGO KOMITETU AUTOMATYCZNEGO PRZETWARZANIA INFORMACJI NACZELNEJ ORGANIZACJI TECHNICZNEJ

THANASIS KAMBURELIS

ELWRO
Wrocław

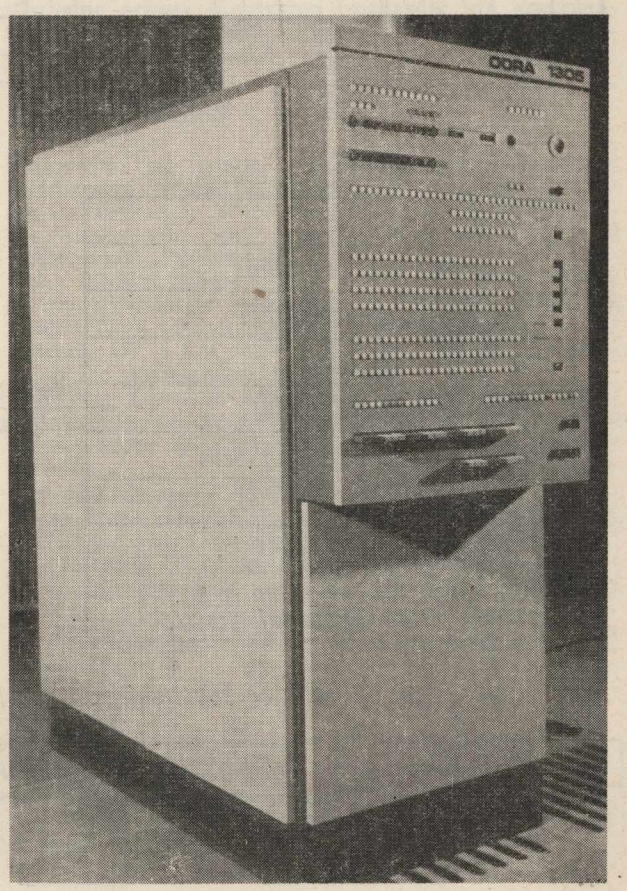
681.322.004.1

System ODRA 1305
Organizacja logiczna i podstawowe dane użytkowe

W artykule podano ogólną charakterystykę komputera trzeciej generacji — ODRA-1305, przeznaczonego głównie do przetwarzania danych w dużych ośrodkach obliczeniowych oraz do obliczeń naukowo-technicznych. Nowy komputer został opracowany wspólnie przez Zakłady ELWRO we Wrocławiu i Instytut Maszyn Matematycznych w Warszawie.

1. Wprowadzenie

Komputery rodziny ODRA znane są powszechnie już od 10 lat. W pierwszych latach pojawienia się komputerów ODRA, główne starania Zakładów ELWRO były skierowane na opracowanie i produkcję dobrego sprzętu technicznego (hardware). Lecz jednostronność tej tendencji została w porę zauważona i od wielu lat problemom opracowania, oprogramowania (software) oraz wyposażenia komputerów ODRA we wszechstronne i nowoczesne oprogramowanie (systemowe i użytkowe) poświęca się najwięcej troski i nakładów finansowych w Zakładach ELWRO. Pierwszym objawem tej troski Zakładów ELWRO było opracowanie i produkcja komputerów ODRA 1304. ODRA 1304 jest powszechnie uważana przez użytkowników za jedną z najlepiej oprogramowanych maszyn w Europie. Komputer ODRA 1304 oparty na technice dyskretnej zapoczątkował całą rodzinę komputerów pod nazwą SYSTEM ODRA 1300. Dziś SYSTEM ODRA 1300 został rozszerzony o bogaty zestaw sprzętu komputerowego, oparty na technice scalonej. Sprzęt ten współpracuje z nowymi jednostkami centralnymi pod nazwą ODRA 1305 i ODRA 1325. Centralna jednostka przetwarzania ODRA 1305, opracowana wspólnie przez ELWRO i Instytut Maszyn Matematycznych, jest obecnie najszybszą i najbardziej wszechstronnie rozbudowywalną maszyną Systemu ODRA 1300. Oprogramowanie maszyny ODRA 1305 jest również bogate i nowoczesne. Ponadto komputer ODRA 1305 (jako największy) akceptuje w pełni oprogramowanie maszyn mniejszych ODRA 1304 i ODRA 1325. Akceptacja oprogramowania jest przy tym bardzo efektywna, gdyż odbywa się na poziomie instrukcji maszynowych. Zatem wieloletni i bogaty dorobek zastosowań (krajowych i zagranicznych), sprawdzony na maszynach ODRA 1304, zostaje automatycznie przeniesiony do maszyny ODRA 1305. Wy-



konanie programów odbywa się około 12 razy szybciej w procesorze ODRA 1305 (wyposażonym w pamięć operacyjną o cyklu 1 mikrosekundy) niż w procesorze ODRA 1304. Bogactwo oprogramowania komputera ODRA 1305 rośnie również dzięki temu, że architektura logiczna komputera ODRA 1305 jest zgodna (kompatibilna) z architekturą logiczną znanych powszechnie komputerów 1905E, F i 1904A firmy International Computers Limited, z którą to firmą Zakłady ELWRO mają odpowiednie porozumienie w sprawie wzajemnego przekazywania software'u.

2. Ogólna charakterystyka komputera ODRA 1305

ODRA 1305 jest komputerem trzeciej generacji, przeznaczonym głównie do przetwarzania danych w dużych ośrodkach obliczeniowych i organizacjach gospodarczych oraz do obliczeń naukowo-technicznych. Jest również przygotowany do sterowania szybkimi procesami zachodzącymi w czasie realnym, dzięki wbudowaniu specjalnego stanu priorytetowego w procesorze. W praktyce ODRA 1305 może wykonywać jednocześnie różne zadania z wyżej podanych dziedzin zastosowań, dzięki posiadaniu takich cech technicznych jak: jednoczesność pracy bloków funkcjonalnych, wieloprogramowość, wielodostępność, dwuprocesowość.

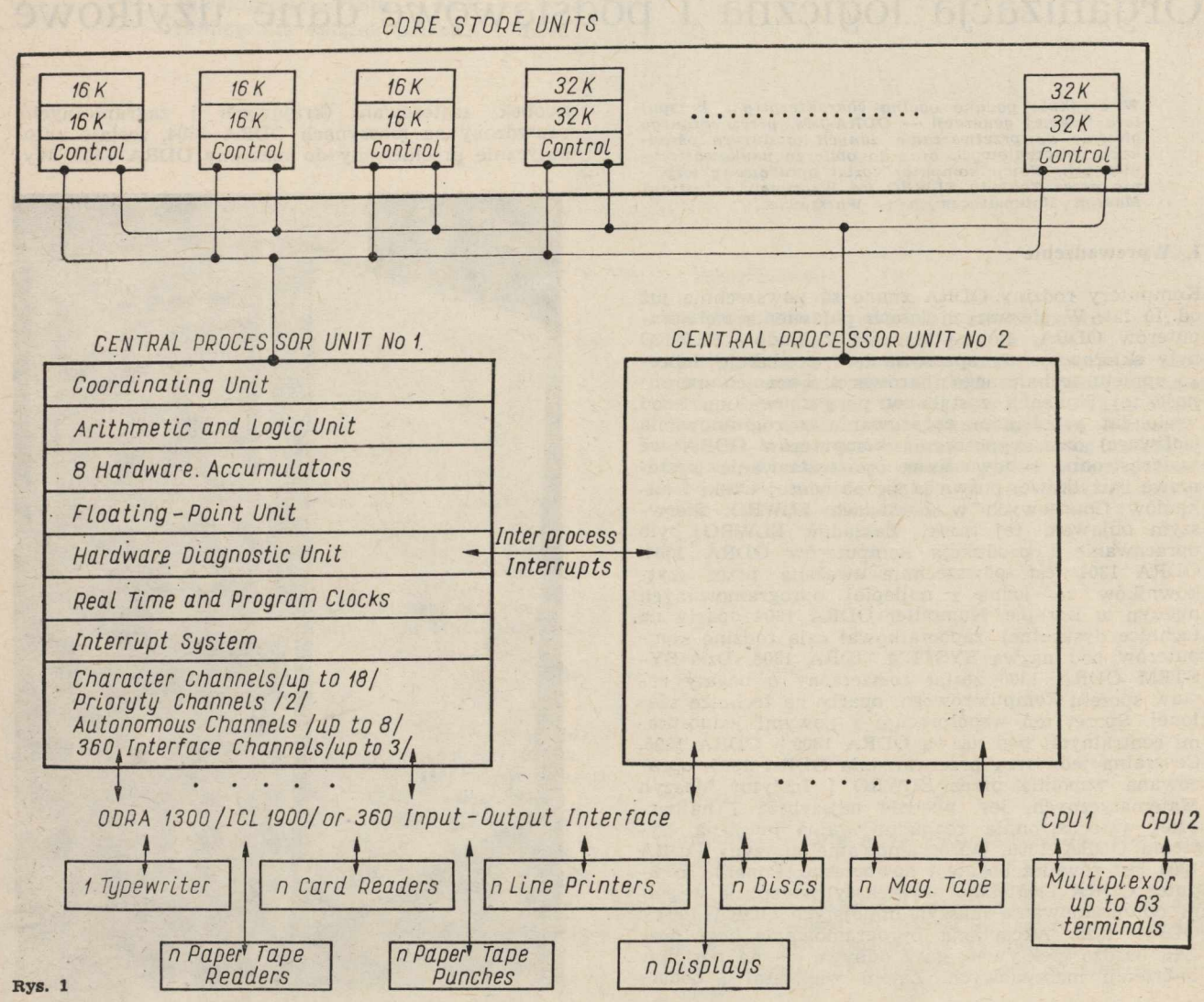
Komputer ODRA 1305 został tak zaprojektowany, aby użytkownik mógł dowolnie, w zależności od swoich potrzeb, dobierać niezbędną dla niego konfigurację sprzętu (wielkości pamięci operacyjnej, liczba i typy kanałów, liczba procesorów, ilość i typy urządzeń zewnętrznych itp.). Stosownie do wybranej konfiguracji sprzętu generuje się software systemowy, gdyż struktura software'u jest również modularna.

Nowoczesne rozwiązania logiczne i konstrukcyjno-technologiczne nadają komputerowi ODRA 1305 następujące cechy:

- Pełną zgodność funkcjonalną i programową z systemami ICL 1905E, F (lub 1904A), dzięki czemu bogate i nowoczesne oprogramowanie komputerów ICL funkcjonuje w komputerze ODRA 1305 bez dokonania jakichkolwiek zmian. Również dowolne urządzenia zewnętrzne serii ICL 1900 mogą być bezpośrednio dołączone do komputera ODRA 1305, dzięki przyjęciu identycznego Interface'u wejścia-wyjścia.

- Pełną zgodność programową (z wyjątkiem oprogramowania technicznego) i Interface'u wejścia-wyjścia z komputerami serii ODRA 1300 (np. ODRA 1304, 1325), dzięki temu bogate oprogramowanie użytkowe ODRA 1304 funkcjonuje w ODRA 1305. oraz szeroki i istniejący zestaw urządzeń zewnętrznych serii ODRA 1300 może być bezpośrednio podłączony do ODRA 1305.

- Dużą elastyczność strukturalną i programową w tworzeniu dowolnych konfiguracji użytkowych.
- Wieloprogramowość (do 16 programów głównych, każdy z 3 subprogramami).
- Wielodostępność: komputer może być wykorzystywany jednocześnie przez 60 użytkowników.
- Dwuprocesorowość: w takim systemie dwa procesory dzielą między sobą wspólną pamięć operacyjną.
- Wysoką niezawodność pracy systemu oraz dużą łatwość lokalizacji ewentualnych uszkodzeń, dzięki wbudowaniu specjalnego sprzętu dynamicznej detekcji i lokalizacji błędów.



Rys. 1

● Dynamiczną rekonfigurację sprzętu (np. w wypadku uszkodzenia danego bloku funkcjonalnego).

● Wysoką wydajność systemu (duża szybkość przetwarzania procesorów, bogata i wydajna lista rozkazów hardware'owo realizowanych, hardware'owe akumulatory ogólne i zmiennoprzecinkowe, szybkie kanały znakowe, szybkie kanały autonomiczne i priorytetowe, interleaving bloków pamięci operacyjnej, jednoczesność pracy wszystkich bloków funkcjonalnych).

Ogólny schemat blokowy komputera ODRA 1305 przedstawia rys. 1.

3. Organizacja logiczna komputera ODRA 1305

ODRA 1305 została tak zaprojektowana, aby praca jej odbywała się pod kontrolą programu sterującego (znanego pod nazwą EXECUTIVE). Program sterujący EXECUTIVE traktowany jest jako integralna część sprzętu technicznego ODRA 1305. Znajduje się on w początkowej części pamięci operacyjnej i jest zabezpieczony przed naruszeniem jego obszaru przez którykolwiek z szesnastu programów użytkowych. Podstawowym zadaniem programu EXECUTIVE jest wykonywanie zleceń operatora lub programisty, sygnalizacja błędów i organizacja efektywnej pracy programów użytkowych (w systemie wieloprogramowym lub wielodostępnym).

EXECUTIVE spełnia swoje zadanie przez wykonanie następujących funkcji:

— sankcjonuje, zapoczątkowuje i kontroluje wszystkie przesłania danych zlecone przez programy użytkowe,

— wykonuje wszystkie funkcje rozkazów ekstrakodowych (tj. rozkazów programowanych),

— kontroluje, wykonuje lub zapoczątkowuje wszystkie czynności komunikowania się z operatorem za pośrednictwem monitora,

— organizuje i kontroluje przebieg pracy wieloprogramowej i wielodostępnej.

Ogólny (przykładowy układ programu EXECUTIVE i programów użytkowych (problemowych) w pamięci operacyjnej pokazuje rysunek 2.

Programy użytkowe zajmują dowolne pole pamięci operacyjnej i są przechowywane w dowolnej kolejności. Łączna długość wszystkich programów nie może przekroczyć liczby 262144 komórek pamięci operacyjnej (czyli 1048576 znaków alfanumerycznych). Adresacja górna rysunku 2 jest absolutną adresacją komórek pamięci operacyjnej zajętych przez poszczególne programy, zaś adresacja dolna jest adresacją względną (relatywną). Programy są przechowywane w pamięci operacyjnej w adresacji względnej, a w czasie wykonywania programu zamienia się automatycznie adresy względne na bezwzględne. Dzięki temu można w razie potrzeby (np. w czasie skreślenia jakiegoś programu) szybko przenosić programy użytkowe z jednego pola pamięci operacyjnej na inne pole bez potrzeby nowej kompilacji programu.

3.1. Formaty informacji

ODRA 1305 wykonuje działania arytmetyczne i logiczne na argumentach 24- lub 48-bitowych oraz działania na znakach i polach składających się z 6-bitowych znaków.

Podstawowe słowo maszyny ma 24 bity. Słowo zajmuje jedną komórkę pamięci operacyjnej lub jeden rejestr procesora. Dwa takie kolejne słowa tworzą tzw. argumenty długie (48-bitowe), zaś z kolejnych słów tworzy argumenty praktycznie dowolnej długości. Słowo 24-bitowe zawiera 4 adresowane znaki (tzw. 6-bitowe byte'y), dla których wbudowano bogaty zbiór operacji maszynowych (np. pobranie znaku alfanumerycznego, pamiętanie znaku, konwersja znaków z układu dziesiętnego na układ binarny i na odwrót, przenoszenie pola znaków itp.).

Rozkaz (instrukcja) ODRA 1305 ma również 24 bity i zajmuje jedną komórkę pamięci operacyjnej. W rozkazie umieszcza się informacje określające:

— typ wykonanej operacji (7-bitowa część F),

— adres jednego z 8 akumulatorów **X** (w którym przechowuje się pierwszy argument operacji), zbudowanych na układach scalonych,

— adres pierwotny komórki pamięci operacyjnej (12-bitowa część N),

— adres modyfikatora M (tj. akumulator X1, X2 lub X3).

Adres N oraz zawartość modyfikatora M (18 bitów) określają tzw. adres efektywny komórki (zawierającej drugi argument operacji) w sposób następujący N:=N+(M); gdzie (M) oznacza zawartość rejestru modyfikatora. Adresy efektywne można również określać różnymi premodyfikacjami i modyfikacjami relatywnymi (tj. według stanu licznika rozkazów). Dokładną strukturę rozkazów przedstawia rys. 3.

3.2. Rejestry i indykatory wewnętrzne

W jednostce centralnej ODRA 1305 znajdują się różne rejestry biorące udział przy wykonywaniu operacji arytmetycznych, logicznych i organizacyjnych oraz wskaźniki (indykatory) opisujące stany poszczególnych bloków funkcjonalnych. Poniżej opiszemy tylko najważniejsze z nich (patrz rysunek 4).

● Rejestry DATUM (DT) i LIMIT (LM). Są to formalnie 18-bitowe rejestry, w których przechowuje się adresy graniczne programu bieżącego. Rejestr DATUM zawiera adres bezwzględny pierwszej komórki pamięci operacyjnej danego programu, zaś rejestr LIMIT adres ostatniej komórki (powiększony o jeden) tego programu. W czasie pracy wszelkie bezwzględne adresy efektywne (rozkazów lub argumentów operacji) programu aktualnie pracującego powinny spełniać nierówność:

$$\text{DATUM} \leqslant \text{ADRES} < \text{LIMIT}$$

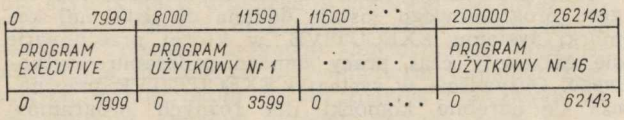
Pojawienie się adresu nie spełniającego powyższej nierówności powoduje przerwanie bieżącego programu i przejście do programu EXECUTIVE. W ten sposób zapewniona jest protekcja programów (przed wzajemnym zniszczeniem informacji w nich zawartych). Każdy program ma własne graniczne adresy przechowywane w odpowiednich tablicach programu EXECUTIVE. Program EXECUTIVE wpisuje w rejestrach DT i LM adresy graniczne danego programu przed powołaniem go do pracy.

● Rejestr Licznika Rozkazów LR (Instruction counter). Rejestr ten zawiera 18 pozycji; przechowuje się w nim adres bezwzględny kolejnego rozkazu do wykonania.

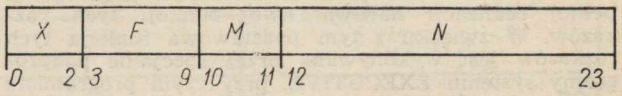
● Rejestr rozkazu składa się z części:
X dla pamiętania adresu akumulatora ogólnego (0÷7),
F dla pamiętania funkcji rozkazu (000÷177),
M dla pamiętania adresu modyfikatora (1÷3),
N dla pamiętania adresu argumentu operacji (0÷4095).

● Akumulator zmiennoprzecinkowy A (48 bitów).

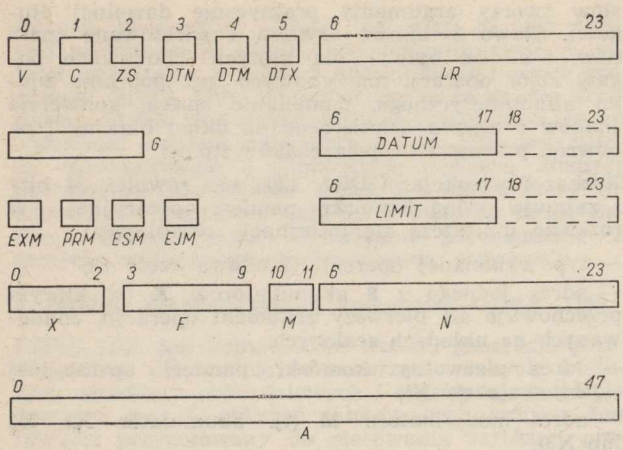
● Wskaźnik nadmiaru V i wskaźnik przeniesienia C.



Rys. 2



Rys. 3



Rys. 4

Wskaźniki EXM (Executive Mode) i PRM (Priority Mode) opisujące stan programowy, w którym znajduje się w danej chwili procesor:

EXM	PRM	stan procesora
0	0	stan programu Normalnego
0	1	stan programu Priorytetowego
1	0	stan programu EXECUTIVE
1	1	stan programu Priorytetowego

Zatem procesor może znajdować się w jednym z trzech stanów. Przechodzi on w stan programu Priorytetowego, gdy otrzyma specjalne przerwanie od tzw. Kanału Priorytetowego (współpracującego bezpośrednio z urządzeniami sterującymi danym procesem przemysłowym zachodzącym w czasie realnym). Przerwania priorytetowe są przyjmowane zarówno w stanie Normalnym jak i w stanie EXECUTIVE. W stanie EXECUTIVE procesor wykonuje różne funkcje (np. inicjowanie przesyłań u urządzeń zewnętrznych, zmiana stanu rejestrów DT i LM), które nie są wykonywane w innych stanach, i traktowane są jako nielegalne (powodujące przerwania programowe).

Wskaźniki ESM (Extended Store Mode) i EJM (Extended Jump Mode); uruchamiają one dodatkowe mechanizmy w procesorze w przypadku, gdy pojemność danego programu przekracza 32 K słów maszynowych (K = 1024).

Rejestr G (4-bitowy), pamięta się w nim numer binarny aktualnie pracującego programu głównego.
Wskaźniki ZS, DTN, DTM i DTX. Wskaźnik ZS służy do likwidacji zer początkowych liczb dziesiętnych otrzymanych przy pomocy specjalnych rozkazów konwersji z układu binarnego na układ dziesiętny. Natomiast wskaźniki DTN, DTM i DTX powodują, gdy ich stan równa się 1, że adresy N, M i X występujące w rozkazach programu EXECUTIVE są automatycznie powiększane o zawartość rejestru DT, czyli w tych przypadkach EXECUTIVE kontaktuje się z komórkami programu użytkowego (o adresach: N+DT, M+DT, X+DT).

3.3. Lista rozkazów

Lista rozkazów (instrukcji) maszyny ODRA 1305 jest bogata i efektywna. Zawiera ponad 140 rozkazów logicznych, arytmetycznych, zmiennoprzecinkowych, konwersji, działań na znakach i na polach z danymi.

3.4. Technika rozkazów ekstrakodowych

Technika ta polega na tym, że pewna grupa rozkazów (tj. rozkazy o kodach 140÷147, 150÷157, 160÷167 i 170÷177 w stanie programu Normalnego) nie ma pełnej realizacji hardware'owej funkcji tych rozkazów. W związku z tym podstawowa funkcja tych rozkazów jest wykonywana przez specjalne podprogramy systemu EXECUTIVE, przy czym programista nie jest świadomy tego faktu. Pojawienie się takiego rozkazu ekstrakodowego powoduje automatyczne przejście do systemu EXECUTIVE, wykonanie funkcji rozkazu przez odpowiedni podprogram w EXECUTIVE i powrót do programu użytkowego.

3.5. Akumulatory hardware'owe

Każdy program użytkowy ma 8 rejestrów 24-bitowych, zwanych akumulatorami ogólnymi X(X0÷X7). W rejestrach tych przechowuje się na ogół jeden z argumentów operacji oraz rezultat operacji. Ponadto każdy program ma jeden 48-bitowy akumulator zmiennoprzecinkowy A. Z punktu widzenia programisty akumulatory ogólne X zajmują 8 pierwszych komórek pamięci operacyjnej danego programu użytkowego (o adresach 0, 1, ..., 7). Zatem każdy z 16 programów użytkowych ma swoje własne akumulatory X i A. Natomiast z punktu widzenia technicznego w procesorze ODRA 1305 istnieje jeden zestaw 8 ogólnych akumulatorów hardware'owych HX(HX0÷HX7) oraz jeden hardware'owy akumulator zmiennoprzecinkowy HA. Akumulatory HX i HA spełniają rolę akumulatorów X i A programu bieżącego. W związku z tym, gdy dany program zostaje uruchomiony, przepisuje się stan akumulatorów X i A tego programu do rejestrów roboczych HX i HA (przepisanie to odbywa się jednym rozkazem). Zatem program bieżący w czasie swej pracy nie musi kontaktować się z komórkami pamięci operacyjnej lecz bezpośrednio z akumulatorami hardware'owymi. Jest to więc ogromna oszczędność czasu, gdyż dostęp do komórki pamięci operacyjnego wymaga czasu około 1 μsek (lub 2 μsek — zależnie od cyklu stosowanej pamięci), zaś dostęp do rejestru hardware'owego jest praktycznie natychmiastowy. Natomiast w czasie przerwania programu bieżącego stan akumulatorów hardware'owych HX i HA zostaje automatycznie przenoszony do komórek o adresach 0÷7 i 12÷13 tego programu; zaś do rejestrów HX i HA ładuje się stan akumulatorów X i A nowego programu.

3.6. Zegar czasu realnego i zegar programowy

W dużych i średnich systemach komputerowych, eksploatowanych jednocześnie przez wielu użytkowników (np. w systemie wieloprogramowym lub wielodostępnym), powstaje problem liczenia czasu zajętości procesora przez poszczególne programy użytkowe. Potrzebny jest również dokładny protokół z przebiegu pracy programów w czasie realnym. W tym celu wybudowano w procesorze ODRA 1305 dwa zegary elektroniczne:

— zegar czasu realnego i

— zegar programowy.

Zegar czasu realnego pracuje „non stop" i odmierza odcinki czasowe równe 200 milisekundom. Oznacza to, że gdy mija kolejny 200-milisekundowy odcinek czasu, zegar powoduje przerwanie programowe i wejście do systemu EXECUTIVE. EXECUTIVE na tej podstawie aktualizuje swoje programowe zegary, w których rejestruje kolejne sekundy, minuty i godziny. Na przykład fakt upłynięcia kolejnej minuty jest sygnalizowany na wydruku monitora, zaś upłynięcia 5 sekund — powoduje pewne kontrole prawidłowości przesyłań danych.

Zegar programowy natomiast służy do dokładnego zliczania czasu pracy poszczególnych programów użytkowych. Jest to rejestr hardware'owy, do którego dodaje się 1 co 4,8 mikrosekundy, gdy funkcjonuje program użytkowy (pojemność hardware'owego zegara programowego wystarcza na 10 sekund). Zatem w zegarze tym akumuluje się czas pracy programu bieżącego z dużą dokładnością. Gdy dany program zostaje przerwany, to zawartość hardware'owego zegara programowego zostaje dodana do ustalonej komórki systemu EXECUTIVE, w której przechowuje się całkowity czas pracy danego programu użytkowego. Oczywiście w systemie EXECUTITVE przeznacza się odrębne komórki dla różnych programów użytkowych. Na tej podstawie wiadomy jest podział

czasu pracy procesora pomiędzy programami użytkowymi.

3.7. System przerwań programowych

W czasie wykonywania programów procesor ODRA 1305 może przyjmować różne sygnały pochodzące od urządzeń zewnętrznych lub od zespołów wewnętrznych procesora (np. przepełnienie zegara czasu realnego, operacja nielegalna, próba naruszenia granic programu bieżącego). Sygnały od urządzeń zewnętrznych pojawiają się na ogół wtedy, gdy dane urządzenie zrealizowało zleconą transmisję danych lub gdy zaszło jakieś nieoczekiwane zdarzenie.

Sygnały te dzielą się na dwie grupy:

— sygnały normalne (od 1 do 63) i

— sygnały priorytetowe (jeden sygnał od jednego kanału priorytetowego).

Sygnały priorytetowe mają pierwszeństwo przed sygnałami normalnymi, również między sygnałami danej grupy istnieje ustalony priorytet obsługi tych sygnałów. Normalne sygnały przerwań są przyjęte tylko wówczas, gdy procesor wykonuje jakiś program użytkowy. Pojawienie się takiego sygnału powoduje przerwanie programu bieżącego i wejście do systemu EXECUTIVE, który zajmuje się ustaleniem przyczyny przerwania programu oraz obsługą tego przerwania (np. organizuje dalsze przesyłanie danych do pamięci dyskowej). Jeśli procesor znajduje się w stanie programu EXECUTIVE, to przerwania normalne muszą poczekać do momentu powrotu do programu użytkowego. Natomiast priorytetowe sygnały przerwań są przyjmowane zarówno w stanie programu Normalnego jak i w stanie programu EXECUTIVE. Po przyjęciu takiego sygnału zapala się wskaźnik PRM i następuje automatyczne przejście do tzw. programu Priorytetowego, który zajmuje się obsługą tego przerwania.

Poniżej podaje się dla przykładu funkcję przerwania normalnego:

— Zapamiętanie stanu akumulatorów hardware'owych w komórkach (programu przerwanego) o adresach DT+0, DT+7.

—Zapamiętanie adresu względnego następnego rozkazu (programu przerwanego), tj. LR+1 z różnymi wskaźnikami (V, C, ZS) w komórce DT+8.

— Zapamiętanie hardware'owego akumulatora zmiennoprzecinkowego w komórkach DT+12 i DT+13.

— Ustawienie wskaźnika EXM w stan 1.

— Zerowanie wskaźników V, C, ZS i rejestru G.

— Wpisanie liczby 16 do licznika rozkazów LR i przejście do wykonania rozkazu z komórki 16 programu EXECUTIVE.

Uwaga: powyższa pełna funkcja przerwania programowego jest wykonywana tylko dla pierwszego przerwania (o najwyższym priorytecie) zaś sygnały dalszych przerwań są (z inicjatywy systemu EXECUTIVE) przyjmowane i obsługiwane przed pełnym powrotem do jakiegoś programu użytkowego.

3.8. System wejścia-wyjścia (WE-WY)

Transmisja danych z urządzeń zewnętrznych do pamięci operacyjnej (i na odwrót) odbywa się za pośrednictwem kanałów.

W maszynie ODRA 1305 wyróżnia się dwa typy kanałów:

— kanały znakowe (w tym multiplexorowe)

— kanały autonomiczne (słowowe).

Przez kanały znakowe podłącza się urządzenia wolne (np. czytnik kart, drukarka wierszowa) zaś przez kanały autonomiczne urządzenia szybkie (np. pamięć taśmowa, pamięć dyskowa). Maksymalna szybkość przesyłania danych za pośrednictwem kanałów znakowych wynosi około 200 000 znaków 6-bitowych na sekundę, zaś kanałów autonomicznych około 500 000 znaków na sekundę. Przy tym kanały autonomiczne nie tylko mają większą szybkość przesyłania lecz także powodują mniejszą zajętość pamięci operacyjnej. Na przykład zapis (lub odczyt) jednego znaku z urządzenia zewnętrznego (przez kanał znakowy) do pamięci operacyjnej zajmuje czas równy (w przybliżeniu) trzem cyklom pamięci, natomiast zapis jednego słowa (tj. czterech znaków na raz) przez kanał autonomiczny zajmuje czas tylko jednego cyklu pamięci, czyli kanały autonomiczne, które są stosunkowo drogie, zajmują pamięć operacyjną (wówczas nie mają dostępu do pamięci inne kanały ani arytmometr) około 12 razy mniej niż kanały znakowe.

W procesorze ODRA 1305 można wbudować do 18 kanałów znakowych oraz do 8 kanałów autonomicznych.

Oprócz kanałów znakowych i autonomicznych, które realizują blokowe transmisje, istnieje także specjalny kanał zwany priorytetowym, do sterowania pracą urządzeń w czasie realnym.

3.8.1. Struktura słów sterujących transmisją

Wszystkie informacje potrzebne do sterowania transmisją danych są zawarte w tzw. Słowie Sterującym Kanału (SSK).

Informacja SSK zajmuje dwie kolejne komórki (programu EXECUTIVE): 256+4K+0 i 256+4K+1 lub 256+4K+2 i 256+4K+3, gdzie K jest numerem bezwzględnym kanału (urządzenia). W komórkach 256+4K+0 i 256+4K+1 pamięta się informacje bieżące transmisji (bieżące SSK), zaś w komórkach 256+4K+2 i 256+4K+3 pamięta się dodatkowe SSK, które opisują transmisję następnego bloku danych. Strukturę słowa sterującego kanału przedstawia rys. 5.

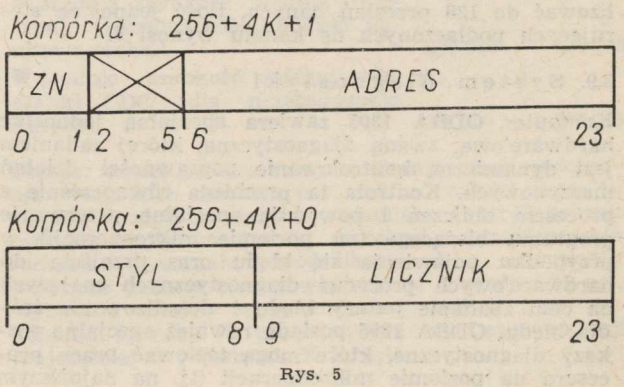
18-bitowa część ADRES określa początkowy adres przesyłania (tj. adres pierwszego przesłanego słowa), zaś 2-bitowa część ZN — określa adres przesyłanego znaku (w przypadku transmisji znakowej, a w przypadku transmisji słowowej część ZN jest obojętna). W czasie transmisji słowowej część ADRES jest powiększana o jeden po przesłaniu jednego słowa, czyli ADRES′ = ADRES+1; zaś część ZN zmienia się (tylko w czasie transmisji znakowej) według następującego schematu:

$$ZN' = ZN+1 \text{ dla } ZN = 0,1,2; \text{ lub}$$

$$ZN' = 0 \text{ i } ADRES' = ADRES+1 \text{ dla } ZN = 3.$$

Część LICZNIK (15-bitowa) podaje liczbę słów lub znaków do przesyłania i jest pomniejszana o jeden po przesłaniu jednego słowa lub znaku. Czyli LICZNIK′ = LICZNIK—1.

Część STYL (9-bitowa) określa bliżej typ (lub sposób) transmisji. Na przykład: wprowadzenie lub wyprowadzenie danych, przesyłanie znaków lub słów, przesyłanie typu „w przód" lub typu „w tył", kanał multiplexorowy lub selektorowy, przesyłanie zwykłe lub specjalne (z automatycznym przepisywaniem sta-



Rys. 5

nu dodatkowego SSK do komórek bieżącego SSK, gdy licznik bieżącego SSK osiągnie stan 0).

3.8.2. Zapoczątkowanie przesyłania danych

Gdy dany program użytkowy żąda wprowadzenia (wyprowadzenia) bloku znaków lub słów z określonego urządzenia pamięci operacyjnej, to stosuje się rozkaz 157 (PERI), który jest rozkazem ekstrakodowym (czyli powoduje skok do EXECUTIVE). W polu sterującym, określonym adresem względnym N rozkazu 157, znajdują się wszelkie informacje potrzebne do zapoczątkowania przesyłania do lub z określonego urządzenia zewnętrznego. Jeśli rozpatrywane urządzenie jest wolne, to EXECUTIVE montuje słowo sterujące kanału SSK na podstawie zawartości komórek N+2 i N+3 pola sterującego rozkazu 157. Ponadto na podstawie komórki N-tej pola sterującego ładuje (za pomocą rozkazu 174E) parametr „sposób przesyłania" do rejestru specjalnego urządzenia zewnętrznego i część STYL słowa SSK, po czym daje sygnał zapoczątkowania przesyłania.

Po tych czynnościach transmisja zostaje zapoczątkowana rozkazem 174E (E — oznacza, że rozkaz ten jest nielegalny w programie normalnym i może być wykonany jedynie w systemie EXECUTIVE) i przebiega autonomicznie, to jest zarówno w kanale znakowym jak i w kanale autonomicznym, zaś program EXECUTIVE oddaję sterowanie (rozkazem 173E) programowi użytkowemu, który zainicjował rozkaz ekstrakodowy 157.

3.8.3. Koniec przesyłania danych

Gdy dana transmisja bloku danych zostanie zakończona, to jednostka sterująca urządzenia wysyła do jednostki centralnej sygnał przerwania (interrupt) programu, oznaczony przez PINT. Sygnał ten jest zarejestrowany w jednej z pozycji rejestru przerwań. Wówczas następuje przerwanie bieżącego programu i skok do komórki 16 programu EXECUTIVE. Na wstępie EXECUTIVE ustala badając rejestr przerwań numer jednostki sterującej urządzenia, które wysłało sygnał PINT. Znając ten numer EXECUTIVE z kolei bada przyczynę przerwania, czytając rejestry specjalne jednostki sterującej urządzenia. Jeśli przyczyną był koniec transmisji danych, w której nie powstał błąd, to program EXECUTIVE uruchamia program, który czekał na to zakończenie przesyłania danych.

3.8.4. Kanały przesyłania danych

ODRA 1305 może być opcjonalnie wyposażoną w zespół kanałów spełniających zasady interface'u wejścia-wyjścia typu IBM360 lub 370.

W skład zespołu mogą wchodzić trzy kanały:

— kanał typu multiplexor i

— dwa kanały typu selektor.

Kanały typu selektor zawierają tylko po jednej drodze (tzn. mogą wykonywać w danej chwili jedną operację wejścia-wyjścia). Natomiast multiplexor zawiera 128 dróg i tym samym może jednocześnie realizować do 128 przesłań danych. Ilość jednostek sterujących podłączonych do kanału wynosi 8.

3.9. System diagnostyki

Komputer ODRA 1305 zawiera specjalną jednostkę hardware'ową, zwaną diagnostyczną, której zadaniem jest dynamiczne kontrolowanie poprawności działań maszynowych. Kontrola ta przebiega równocześnie z procesem obliczeń i powoduje specjalne przerwanie programu bieżącego (na poziomie mikrooperacji) w przypadku pojawienia się błędu oraz przejście do hardware'owych procedur diagnostycznych mających na celu zbadanie natury błędu i zlokalizowanie źródła błędu. ODRA 1305 posiada również specjalne rozkazy diagnostyczne, które mogą testować pracę procesora na poziomie mikrooperacji (tj. na najniższym poziomie) oraz dokonywać fotografii stanu wewnętrznego procesora na każdym etapie wykonania rozkazu. Stany te mogą być z kolei analizowane przez software'owe systemy diagnostyki (szczególnie przydatne w systemie dwuprocesorowym). Wykrycie błędu trwałego w danym procesorze powoduje również tzw. przerwanie międzyprocesorowe, dzięki któremu całość pracy zostaje automatycznie przejęta przez drugi procesor (łącznie z funkcją stawiania diagnozy dotyczącej błędu uszkodzonego procesora). W ten sposób bezawaryjność dwuprocesorowego systemu liczącego ODRA 1305 jest bardzo duża.

W maszynie ODRA 1305 wbudowano dwa typy dynamicznej detekcji błędów:

— detekcja ciągła, towarzysząca operacjom: przesyłań, logicznym i arytmetycznym wykonywanym w procesorze oraz operacjom przesyłań wykonywanym w kanałach znakowych i autonomicznych,

— detekcja okresowa, zachodząca kilkanaście razy w czasie jednej sekundy, przeprowadza kompletną kontrolę poprawności pracy procesora za pomocą optymalnie dobranych szablonów testowych oraz kompletną kontrolę kanałów autonomicznych (to jest wszystkich dróg przesyłania, rejestrów adresów i liczników oraz półsumatorów kanału).

Błędy wykryte przez układy detekcji kanałów (nie zawsze oznacza to, że źródło błędu znajduje się w kanale, bowiem najczęściej błędy powstają w urządzeniach zewnętrznych lub na liniach interface'u) są sygnalizowane jednostkom sterującym urządzeń zewnętrznych, gdzie zostają zarejestrowane i powodują przerwania programowe (to jest wejście do systemu EXECUTIVE). System EXECUTIVE powtarza w takich przypadkach przesłanie bloku danych kilkanaście razy. Jeśli błąd zniknie, to zostanie on zakwalifikowany jako losowy (i odpowiednio zarejestrowany), a praca systemu i urządzenia będzie kontynuowana.

3.10. System dwuprocesorowości

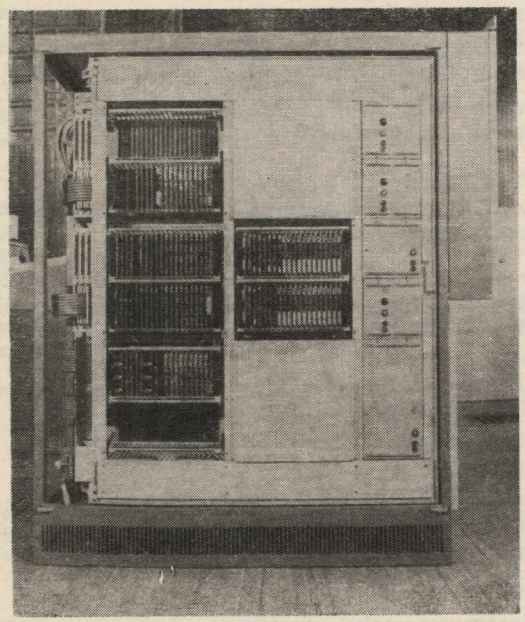
W dużych lub średnich systemach obliczeniowych jednym z najdroższych elementów systemu jest pamięć operacyjna (główna), przy czym czas wykorzystania pamięci operacyjnej (zwłaszcza niezależnych bloków) przez procesor, pracujący w systemie jednoprocesorowym, jest tylko małym ułamkiem całkowitego czasu obliczeń. Wykorzystanie bloków pamięci operacyjnej może być zwiększone, jeśli pamięć operacyjna zostanie tak zorganizowana, aby była wspólną pamięcią dla dwóch lub więcej procesorów i w tym tkwi głównie koncepcja wieloprocesorowości. Ważnym celem wieloprocesorowości jest także zwiększenie niezawodności systemu liczącego. Uszkodzenie dowolnego bloku pamięci operacyjnej lub procesora nie uniemożliwia pracy systemu (chociaż wydajność obliczeniowa systemu maleje w takich przypadkach). Procesory ODRA 1305 są tak budowane, aby można je było zestawiać w system dwuprocesorowy bez żadnych zmian (natomiast po małych zmianach także w system czteroprocesorowy). Również niezależnie (logicznie) bloki pamięci operacyjnej mają odrębne drogi (szyny) współpracy z każdym procesorem. System dwuprocesorowy można również łatwo rekonfigurować przez zmniejszenie (lub zwiększenie) pojemności pamięci operacyjnej lub przez tworzenie dwóch niezależnych systemów jednoprocesorowych. Oczywiście system dwuprocesorowy wymaga specjalnego systemu operacyjnego (który jest połączeniem funkcji programu EXECUTIVE i GEORGE). System ten będzie sterował pracą obu procesorów i wszystkich urządzeń zewnętrznych (połączonych na ogół do różnych procesorów). Organizacja dwuprocesorowego systemu ODRA 1305 pozwala także na niezależny dostęp do urządzeń zewnętrznych od dwóch różnych procesorów, co daje możliwość szybkiej rekonfiguracji urządzeń zewnętrznych.

3.10.1. Pamięć operacyjna

Pamięć operacyjna (rys. 6) składa się z pewnej liczby niezależnych bloków o pojemności 32K lub 64K

słów 25-bitowych i o łącznej pojemności systemu 256*K*. Każdy blok pamięci operacyjnej współpracuje odrębnymi szynami z każdym z procesorów. Zatem każdy blok pamięci ma dwie kompletne szyny, na których podawane są następujące informacje:

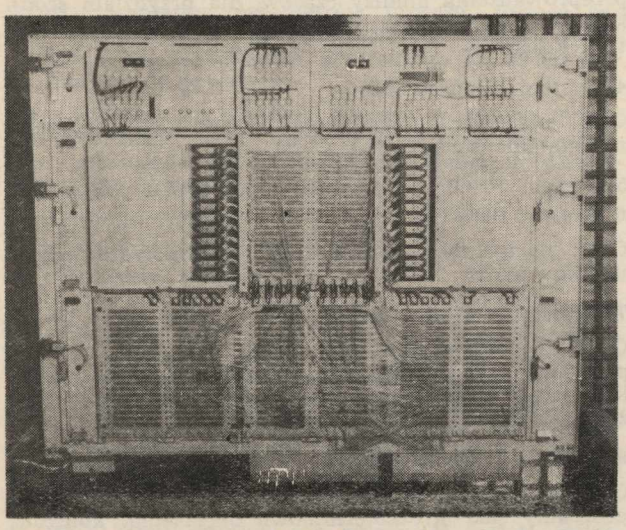
— adres komórki bloku,
— informacja odczytywana z bloku,
— informacja do zapisu w bloku,
— informacje sterujące sposobem współpracy.



Rys. 6

3.10.2. Adresowanie początkowego obszaru pamięci

Cała pamięć operacyjna jest normalnie dostępna dla programów wykonywanych w obu procesorach z wyjątkiem obszaru początkowego o adresach od 0 do 255. Obszar ten jest podzielony na 8 podobszarów, każdy po 32 komórki. Każdemu procesorowi przypisuje się jednoznacznie jeden taki podobszar. Czyli istnieje możliwość, w ogólnej koncepcji, zorganizowania systemu ośmioprocesorowego. Komórki danego podobszaru są normalnie dostępne tylko dla odpowiedniego procesora. W tym obszarze przechowuje się akumulatory pamiętane, różne ślady oraz wejścia związane z przerwaniami danego procesora. Ponieważ każdy procesor adresuje swoje komórki od 0, to wprowadzono w każdym procesorze układ realizujący dodatkową modyfikację: $N+32 \cdot p \cdot d$, dla $N \leqslant 256$; gdzie N oznacza adres skuteczny procesora, p — numer danego procesora, d — parametr mający wartość 1, gdy procesor jest nastawiony na pracę wieloprocesorową, a wartość 0 w przypadku pracy jednoprocesorowej.



Rys. 6a

Poniżej opisuje się jeszcze jeden podział pamięci operacyjnej związany z systemem transmisji. Jak już wiadomo, w systemie jednoprocesorowym komórki początkowe o adresach od $256+4K+0$ do $256+4K+3$ zawierają słowa sterujące kanału o numerze K. W przypadku systemu wieloprocesorowego podłącza się do danego procesora różne urządzenia, podłączone do różnych procesorów mogą mieć przyporządkowane te same numery. W związku z tym, w celu uniknięcia konfliktów, przydziela się każdemu procesorowi odrębny obszar (po 256 komórek), dla przechowania słów sterujących kanałów, według wzoru: $256+4K+256 \cdot p$; gdzie p jest numerem procesora.

3.10.3. Rozkazy pracy wieloprocesorowej

Dla usprawnienia organizacji pracy wieloprocesorowej wprowadzono różne rozkazy wykonywane tylko w stanie EXM=1. Poniżej wymienimy dwa najważniejsze z nich (o kodach 150E i 151E).

Rozkaz 150E (Czytaj i Blokuj)

Czytaj z określonego pola (przez adres N) pamięci operacyjnej oraz pozwól innym procesorom czytać z tego pola także, lecz zabroń (zablokuj) im pisania do tego pola. Rozkaz ten jest wykorzystywany przy ustawianiu zakazu dokonania zmian w danym polu aż do odwołania tego zakazu.

Rozkaz 151E (Pisz i Blokuj)

Pisz do określonego pola (przez adres N) pamięci operacyjnej oraz zablokuj zarówno zapis do tego pola jak i odczyt z tego pola dla innych procesorów. Rozkaz ten jest wykorzystywany przy dokonywaniu korekty informacji danego wspólnego pola i poprzedza sekwencję rozkazów dokonujących korekty tego pola.

4. Podstawowe dane techniczne

Poniżej podaje się najważniejsze dane użytkowe komputera ODRA 1305

czasy wykonania podstawowych operacji (w mikrosekundach) maszyny wyposażonej w minimum 32*K* słów (wówczas istnieje interleaving bloków pamięci operacyjnej, tj. jednoczesna praca bloków niezależnych o pojemności 16*K* lub 32*K*):

● pobranie liczby (dla 1 lub 2 μs cyklu pamięci)	1,2	2
● dodawanie stałoprzecinkowe	1,6	3
● porównanie logiczne	1,6	3
● mnożenie stałoprzecinkowe	9	14
● dzielenie stałoprzecinkowe	14	20
● dodawanie zmiennoprzecinkowe	10	18
● mnożenie zmiennoprzecinkowe	22	40
● skoki	1	2
● średnia szybkość według mieszanki Gibssona (dla obliczeń naukowo-techn.)	280 000	150 000
● średnia szybkość według mieszanki CDC (dla przetwarzania danych)	400 000	220 000

Uwaga: wyżej podane czasy zawierają czasy wykonania wszystkich funkcji tzw. cyklu rozkazowego, tj. wyboru i odczytu rozkazu z pamięci, kontroli adresu rozkazu względem granic protekcji, rozszyfrowania rozkazu i ewentualnej modyfikacji adresu argumentu (wówczas podany czas powiększa się o około 0,25 μs), wyboru i odczytu argumentów operacji (x i *n*), wykonania operacji i przechowania rezultatu operacji (najczęściej w rejestrach X lub *A*).

pamięć operacyjna:

— pojemność ogólna	32*K*, 64*K*, 96*K*, 128*K*, 256*K*
— pojemność samodzielnego bloku	32*K*, 64*K*
— cykl pamięci	1 μs, 2 μs
— czas dostępu	0,4 μs, 0,8 μs
— ilość niezależnych pamięciowych szyn	od 1 do 4

kanały przesyłania danych:

— znakowe	do 18
— multiplexorowe	do 2
— autonomiczne	do 8
— szybkość znakowych	200 000 100 000 zn/s
— szybkość autonomicznych	500 000 500 000 zn/s

— kanały typu 360 : 1 Multiplexor i 2 Selektory

5. Urządzenia zewnętrzne komputera ODRA 1305

W komputerze ODRA 1305 urządzenia zewnętrzne są podłączane do kanałów jednostek centralnych poprzez tzw. Złącze Standardowe, które jest w pełni zgodne ze złączem ODRA 1304, ODRA 1325 i ICL 1900. Do ODRA 1305 można dołączać dowolne ilości urządzeń zewnętrznych z następującego zestawu:

— Monitor typ MO-305-i, szybkość 10 znaków na sekundę.

— Czytnik Taśmy Papierowej CT304-1 lub CT305-1 (wersja scalona), szybkość 1000 znaków na sekundę.

— Perforator Taśmy Papierowej PT304-1 lub PT305-1, 100 znaków na sekundę.

— Czytnik Kart CK304-2 lub CK305, 1000 kart na minutę.

— Drukarka Wierszowa DW304-1 lub DW305-1, 1300 linii na minutę i po 120 znaków w linii.

— Adapter Pamięci Taśmowej APT-304-2, do 6 przewijaków pamięci taśmowej typ PT2.

— Pamięć Taśmowa PT2, szybkość przesyłania około 43 000 znaków 6-bitowych na sekundę.

— Adapter Pamięci Taśmowej APT-305-1, do 6 przewijaków pamięci taśmowej typu PT3.

— Pamięć Taśmowa PT3, szybkość przesyłania do 128 000 znaków na sekundę.

— Pamięć Bębnowa PB-304-1, po 1 048 576 znaków 6-bitowych w każdym module (tj. w czterech jednostkach bębnowych).

— Pamięć Bębnowa PB-305-1, po 2,6 mln znaków 6-bitowych w jednostce bębnowej, szybkość przesyłania 130 000 znaków na sekundę. Do jednej jednostki sterującej można dołączyć do 8 jednostek bębnowych.

— Pamięć Dyskowa ICL 2802 (lub odpowiednik), po 8 mln znaków 8-bitowych w jednostce dyskowej, szybkość przesyłania 208 000 znaków na sekundę.

— Monitor Ekranowy Alfanumeryczny MEA305-1 z buforem na 1040 znaków (26 wierszy po 40 znaków). Do jednej Jednostki Sterującej można dołączyć do 8 monitorów typu MEA305-1.

— Multiplexor MPX1325, który umożliwia dołączenia do 63 wolno pracujących urządzeń (do 200 znaków na sekundę) oraz urządzenia do transmisji danych za pośrednictwem linii telefonicznych i telegraficznych.

— Dowolne urządzenie firmy ICL (Anglia) współpracujące w myśl zasad opisanych przez Standard Interface serii ICL 1900.

6. Oprogramowanie komputera ODRA 1305

Powszechnie wiadomo, że sam zestaw urządzeń techniki cyfrowej jest tylko zbiorem metalowych skrzyń zapełnionych sprzętem elektronicznym. Dopiero bogate i sprawne oprogramowanie czyni te urządzenia przydatnym, wydajnym i nowoczesnym narzędziem pracy umysłowej.

Zakłady ELWRO włożyły wiele wysiłków wspólnie z Instytutem Maszyn Matematycznych oraz z innymi instytutami, uczelniami naukowymi i organizacjami EPD w zapewnienie komputerom ODRA 1300 nowoczesnego i bogatego oprogramowania. Szczególnie wszechstronne oprogramowanie, użytkowe i systemowe mają komputery ODRA 1305.

Oprogramowanie ODRA 1305 jest w pełni zgodne z oprogramowaniem 1905E,F firmy ICL.

Systemy operacyjne:

1. System EXECUTIVE — podstawowy system sterowania pracą maszyny
2. System GEORGE — wyższy system sterowania
3. System MOP — dla sterowania pracą maszyny w systemach wielodostępnych

Języki programowania

1. PLAN — podstawowy i uniwersalny język programowania typu Assembler
2. NICOL — prosty język pracy na kartach perforowanych
3. COBOL — powszechny język wyższego poziomu dla zastosowań ekonomicznych
4. Compact COBOL
5. FORTRAN — uniwersalny język wyższego poziomu dla zastosowań matematycznych
6. Basic FORTRAN (FORTRAN II)
7. ALGOL
8. Basic ALGOL
9. CSL i SIMON — języki do sterowania i symulacji
10. JEAN — język konwersacyjny

Systemy zarządzania i innych zastosowań

W oprogramowaniu użytkowym komputera ODRA 1305 istnieje kilka setek samodzielnych kompletów (pakietów) programów dla różnych zastosowań. Poniżej wymienione są tylko nazwy i ewentualnie krótki opis funkcji niektórych z tych pakietów zastosowań.

1. System PROMPT

Opis: system kierowania i kontroli, w którym występują takie programy jak: zestawienie zapotrzebowania materiałowego, ustalanie ilości materiałów netto i grupowanie ich w branżach, kontrola postępu produkcji, kontrola kosztów, dokumentacja pracy, kontrola zakupów.

2. System SCAN

Opis: kontrola zapasów magazynowych metodą krótkoterminowego przewidywania.

3. System PERT

Opis: pełny zestaw powiązanych programów dla przeprowadzenia analizy czasów, dla przydziału środków, dla planowania wieloprojektowego i dla kontroli kosztów.

4. System Bilansowania

Opis: planowanie i kontrola procesu produkcyjnego metodą bilansowania.

5. System PROP (szacowanie stopy zysku projektów)

6. System PEWTER

Opis: uproszczona wersja PERT dla początkujących użytkowników.

7. System POWER (etap 1)

Opis: system kontroli produkcji obejmujący takie programy jak: analiza przypadków awaryjnych, ustalanie ilości materiałów netto, długoterminowe obciążenia

8. System POWER (etap 2)

Opis: system kontroli produkcji obejmujący takie programy jak: krótkoterminowe obciążenia, kontrola nad zapasami magazynowymi, dokumentacja techno-

logiczna, kontrola zakupów i kontrola postępów produkcji.

9. System PLUTO

Opis: system jest wykorzystany do sterowania działalnością przedsiębiorstwa, może np. kontrolować produkcję, gospodarkę zapasami i narzędziami, kontrolować zakupy, politykę finansową, politykę sprzedaży.

10. Systemy zastosowań przy różnych pracach inżynieryjnych

— System Wytyczania Poziomych Linii Przejścia
— System Wytyczania Poziomych Linii Obwodowych
— Projektowanie i Analiza Kanałów Wodnych
— Analiza Belki Jednorodnej
— Analiza Dwuwymiarowych Ram i Kratownic
— Analiza Ramy Przestrzennej
— Wymiarowanie Rur Układów Zamkniętych
— Zwarcie w Sieciach Trójfazowych
— Rozkład Obciążeń w Systemach Energetycznych
— Prognoza Ruchu Ulicznego
— Programy Wyznaczania Ruchu Ulicznego
— System MILMAP (program przygotowania sterujących taśm papierowych, użytych przy numerycznym sterowaniu frezarek)
— Analiza Systemów Energetycznych

11. Systemy informacyjne i komercjalne

— System NIC (system tworzenia różnych typów indeksów i katalogów)
— System FIND (system odzyskiwania wybranych informacji z kartoteki (files) programowanych na taśmach magnetycznych)
— System Automatycznego Składania Tekstów Drukarskich

12. Systemy matematyczne, statystyczne i badań operacyjnych:

— Optymalne Cięcie Odcinków
— Optymalne Cięcie Prostokątów
— Planowanie Przewozów
— Działania Macierzowe
— Programowanie Liniowe
— Analiza Statystyczna
— Analiza Pomiarowa
— Mieszanka o Najniższym Koszcie

13. Programy i podprogramy biblioteczne
Biblioteka programów i podprogramów standardowych komputera ODRA 1305 składa się sponad 1000 pozycji.

MAREK HOŁYŃSKI
Instytut Maszyn Matematycznych
Warszawa

519.217

Przewidywanie tendencji zmian procesów za pomocą łańcuchów Markowa

Przedstawiono rozwiązanie zagadnienia przewidywania tendencji zmian parametrów procesów przypadkowych, z zastosowaniem rachunku prawdopodobieństwa. Ze względu na dyskretność pomiarów przyjęto schemat szczególnego przypadku procesu Markowa. Podano algorytm, który może być zastosowany do nadzorowania pracy elektrowni, cementowni itp.

Maszyna cyfrowa stosowana w procesach technologicznych wykorzystywana jest na ogół do wykonywania dwóch zadań.

Pierwsze — to przetwarzanie danych, uzyskiwane w wyniku pomiarów parametrów procesu, służące celom statystycznym i sprawozdawczym. Drugim zadaniem są bieżące obliczenia, pozwalające na bezpośrednie sterowanie procesem.

Aby wpływać na przebieg procesu w sposób możliwie optymalny, należy umieć przewidywać jego przyszłe zachowanie. W procesach rzeczywistych rozróżniamy trzy składowe:

a) zdeterminowana
b) przypadkowa
c) „całkowicie" przypadkowa.

W procesie mogą występować wszystkie trzy, dwie lub tylko jedna z tych składowych. W tym ostatnim przypadku proces będzie miał jednolity charakter. I tak, procesami zdeterminowanymi nazwiemy procesy wywołane działaniem określonych przyczyn. Znając działanie każdej z nich potrafimy określić wynik wypadkowy. Sama funkcja precyzująca tę zależność musi być znana.

Jednakże w większości przypadków należy wiedzieć, do jakiej klasy funkcji ona należy. Można ją wtedy ekstrapolować, np. za pomocą ekstrapolatorów automatycznych dyskretnych. Zwiększenie czasu wyprzedzania nie zmniejsza dokładności przewidywania.

Procesy przypadkowe (stochastyczne) są opisywane funkcjami przypadkowymi. Badanie ich oparte jest na obserwowaniu procesu w celu określenia jego statystycznych prawidłowości.

Nie można w tym przypadku sprecyzować przyczynowego związku zjawisk, chociaż związek taki obiektywnie istnieje. Można dokonywać przewidywania, ale tylko dla krótkiego czasu wyprzedzania. Zwiększenie czasu wyprzedzania powoduje bowiem zmniejszenie dokładności predykcji.
W procesach „całkowicie" przypadkowych — w zasadzie nie da się przewidzieć przyszłości.

Interesować się będziemy rozwiązaniem zagadnienia przewidywania tendencji zmian parametrów procesu dla składowej przypadkowej; prościej — parametrów procesów przypadkowych. Na podstawie obserwacji

Mgr inż. Marek Hołyński ukończył w roku 1970 studia na Wydziale Elektroniki Politechniki Warszawskiej. Staż dyplomowy odbywał we Włoszech, zajmując się zagadnieniami automatyzacji w energetyce. Obecnie jest asystentem w Instytucie Maszyn Matematycznych, gdzie pracuje nad problemami syntezy układów scalonych o wysokim stopniu integracji.

tych parametrów, będziemy się starać zatem wyrokować o przyszłym przebiegu procesu.

1. Realizacja przewidywania za pomocą badania pochodnej i funkcji autokorelacji

Najprostszym, nasuwającym się od razu rozwiązaniem tak postawionego problemu jest analiza pochodnej obserwowanego parametru. Pochodna „śledzi" zmiany wartości tego parametru, wskazując na tendencję jego aktualnych zmian. Gwałtowny wzrost lub spadek wartości pochodnej świadczyłby o szybkich zmianach procesu, co stanowiłoby interesujący przypadek dla maszyny.

Zwykle bowiem obserwowane procesy znajdują się w stanach ustalonych i maszyna cyfrowa spełnia rolę nadzorczą, kontrolując ich niezmienność. Gwałtowna zmiana, np. zadanej, stałej temperatury pieca hutniczego może spowodować, iż temperatura ta znajdzie się w zakresie alarmowym i nastąpi awaria.

Celem zwiększenia dokładności przewidywania można wykorzystać drugą pochodną, ale w praktyce wystarcza ograniczenie się do pierwszej.

Jednakże przy analizie pochodnej spod kontroli maszyny wymykały się nieznaczne, długookresowe zmiany parametru. Nie dałyby się one wykryć (małe zmiany pochodnej), a mogłyby doprowadzić parametr do zakresu alarmowego. Zaradzić temu można przez nieco pracochłonne pamiętanie wartości pochodnej wraz z wartością parametru w momencie pomiaru. Ale to z kolei nie skutkuje w przypadku skokowych zmian parametru. Pochodna osiąga wtedy bardzo dużą wartość, a nie świadczy to wcale o przekroczeniu zakresu bezpiecznego przez obserwowany parametr, bo skok ten może być mały. Dla pełnego obrazu sytuacji, trzeba więc też pamiętać wielkość skoku.

Ze względu na to, że zwykle maszyna dokonuje obserwacji co pewien czas (obiegając kolejno wszystkie punkty pomiarowe), możemy rozważać zmiany parametru jako zjawisko dyskretne. Maszyna może obliczać jedynie różnicę między wartościami kolejnych pomiarów oraz pamiętać samą wartość ostatniego pomiaru. Jeśli w wyniku dużego skoku (dużej różnicy) znajdziemy się blisko granicy alarmu, to są duże szanse na to, iż w następnym pomiarze wyjdziemy z zakresu bezpiecznego. Maszyna powinna sygnalizować alarm.

W celu uproszczenia algorytmu, można ustalić wskaźnik wiążący parametry, wpływające na aktualne zagrożenie alarmowe. Można np. ten wskaźnik zaproponować w postaci:

$$m^{(n)} = \frac{x_n - x_{n-1}}{x_{gr} - x_n} \tag{1}$$

gdzie:

x_n — wartość pomiaru n
x_{n-1} — wartość pomiaru n-1
x_{gr} — wartość granicy alarmu

$x_n - x_{n-1}$ obrazowałoby zmiany w przerwie przed ostatnim pomiarem, a $x_{gr} - x_n$ wskazywałoby na odległość zakresu alarmowego.

Począwszy od pewnej wartości m_{gr} należałoby sygnalizować alarm; optymalnie dla $m \geqslant 2$.
Następnym teoretycznym rozwiązaniem, które można by zastosować do przewidywania alarmów, jest badanie funkcji autokorelacji. Wartość funkcji autokorelacji wskazywałaby na zależność między przyszłymi i obecnymi wartościami parametru.

Stanowiłoby to podstawę do oceny prawdopodobieństwa znalezienia się w zakresie alarmowym dla parametru, którego wartość znamy obecnie. Funkcję autokorelacji wyznaczamy następująco:

$$R_{xx}(l) = \frac{l}{N-l} \sum_{k=1}^{N-l} x(k) \cdot x(k+1) \tag{2}$$

(dla wygody przyjmujemy, że $m_{ox} = 0$),
gdzie N jest liczbą pomiarów, w których obliczamy funkcję autokorelacji.

Jednakże funkcję autokorelacji można przedstawić w taki wygodny do obliczeń sposób jedynie wtedy, gdy nie zależy ona od kolejnego punktu pomiarowego k, tzn. tylko dla stacjonarnych procesów przypadkowych.

Chcąc zatem obliczyć funkcję autokorelacji w prosty algorytmicznie sposób, ograniczylibyśmy się do klasy procesów stacjonarnych. Cała analiza pozostałych procesów oparłaby się na przybliżeniu zakładającym stałość charakteru zmienności procesu w długim okresie czasu.

Kolejnym możliwym rozwiązaniem problemu jest badanie wyprzedzania za pomocą procesów Markowa.

2. Łańcuchy Markowa

Przy rozpatrywaniu przebiegu obserwowanego procesu na bieżąco znajduje zastosowanie rachunek prawdopodobieństwa. Kolejne wartości mierzonego parametru można uznać za osobne zdarzenie losowe. Oczywiście nie są to zdarzenia niezależne. Dogodną zatem do analizy wydaje się metoda badania zdarzeń losowych zależnych, oparta na pojęciu procesu Markowa — statystycznego modelu procesu o zmiennej losowej zależnej. Do naszych potrzeb — ze względu na dyskretność pomiarów — nadaje się schemat łańcucha Markowa, tj. szczególnego przypadku procesu Markowa.

Pojęcie łańcucha Markowa można w skrócie (jest ono na ogół znane) wyjaśnić następująco:

Mamy ciąg $n = 0, 1, 2...$ doświadczeń-pomiarów. W wyniku każdego z nich może zaistnieć tylko jedno zdarzenie spośród skończonego lub przeliczalnego zbioru zdarzeń. Zdarzenie takie polega na otrzymaniu w danym pomiarze konkretnej wartości mierzonego parametru. Wartość ta należy do zbioru stanów $X_i = 1, 2, 3... u$, które parametr ten może przejmować. Określamy prawdopodobieństwo przejścia w kolejnym pomiarze z jednego stanu do innego. Np. prawdopodobieństwo tego, że w n-tym pomiarze parametr będzie w stanie x_j, gdy w (n-1) był w stanie x_i, oznaczymy $p_{ij}^{(n)}$;

$$p_{ij}^{(n)} = P(X^n = x_j / X^{n-1} = x_i) \tag{3}$$

Ten ciąg pomiarów jest połączony w łańcuch Markowa, gdy dla dowolnych stanów x_i w X^{n-1} oraz $n = 0,1, 2...$ równość (3) zachodzi dla dowolnych stanów osiągniętych w $X^{n-2} ... X^0$, tzn. w pomiarach n-2, n-3, ..., 0 parametr mógł osiągać dowolne stany i nie wpływa to na $p_{ij}^{(n)}$.
Istotnie w rejestrowanych przez nas w praktyce procesach jest to na ogół spełnione.

Natomiast założenie skończoności i przeliczalności zbioru zdarzeń jest spełnione automatycznie. Zdarzenia te bowiem w naszym przypadku są wskazaniami cyfrowymi, których liczba jest skończona ze względu na konstrukcję fizyczną maszyny.

3. Prawdopodobieństwo przejść

Załóżmy chwilowo, że parametr analizowany jest po pewnym czasie od początku procesu, dostatecznie długim, aby zapamiętany w pamięci maszyny przebieg mógł stanowić podstawę do przewidywań statystycznych. W oparciu o doświadczenie zdobyte w przeszłości można obliczyć wszystkie prawdopodobieństwa przejść z każdego stanu poprzedniego X_{n-1} do każdego następnego X_n:

$$p_{ij}^{(n)} = \frac{v}{r-1} \tag{4}$$

gdzie v jest liczbą przejść ze stanu x_i do stanu x_j w dotychczasowych r-1 pomiarach, w których osiągnęliśmy stan x_i.

Oczywiście możemy obliczyć i inne prawdopodobieństwo dające informację o przebiegu parametru, jak np. prawdopodobieństwo przebywania w konkretnym stanie. Jednakże prawdopodobieństwo przejścia między dwoma kolejnymi stanami jest wygodne do obliczania i łatwiejsze do wykorzystania.

Dla poręczniejszego zapisu, zbierzmy obliczone prawdopodobieństwa w macierz przejść:

$$M_{nn} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & \vdots \\ p_{n1} & \cdots & & p_{nn} \end{bmatrix} \tag{5}$$

Cechą charakterystyczną tej macierzy jest to, że suma prawdopodobieństw w każdym wierszu jest równa jedności. Wiersze są bowiem rozkładami prawdopodobieństwa przejścia.

Po dokonaniu pomiaru musimy znaleźć się w którymkolwiek ze stanów x_j wykluczających się wzajemnie. Przypadek $\sum p_{i=\text{const}\,j} < 1$ należy odrzucić. Oznaczałoby to bowiem zatrzymanie się łańcucha Markowa — zniknięcia parametru. Suma prawdopodobieństw w kolumnie nie musi być równa jedności.

Macierz przejść daje się zastosować praktycznie do przewidywania przyszłych stanów, w szczególności interesujących maszynę stanów alarmowych.
Gdy prawdopodobieństwo przejścia ze stanu, w którym znajduje się proces w danej chwili obliczeniowej do stanu alarmowego jest tak duże, że należy liczyć się z jego możliwością, wtedy trzeba zasygnalizować alarm i przeciwdziałać jego wystąpieniu, uruchamiając odpowiedni program alarmowy.

To, jak duże prawdopodobieństwo P_g powoduje przeciwdziałanie maszyny zależy od naszej decyzji. Podejmujemy ją, biorąc pod uwagę, jak groźne są skutki nieuprzedzenia alarmu przy określonym prawdopodobieństwie jego wystąpienia w następnym pomiarze.

Niech dla przykładu dany będzie proces, którego obserwowany parametr przyjmuje $u = 6$ możliwych stanów. Jego macierz wygląda następująco:

$$M_{(6\times6)} = \begin{bmatrix} 0{,}3 & 0{,}4 & 0{,}2 & 0{,}08 & 0{,}02 & 0{,}0 \\ 0{,}05 & 0{,}25 & 0{,}5 & 0{,}15 & 0{,}05 & 0{,}0 \\ 0{,}0 & 0{,}01 & 0{,}49 & 0{,}49 & 0{,}01 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}3 & 0{,}4 & 0{,}25 & 0{,}05 \\ 0{,}02 & 0{,}08 & 0{,}1 & 0{,}3 & 0{,}3 & 0{,}2 \\ 0{,}0 & 0{,}02 & 0{,}08 & 0{,}2 & 0{,}4 & 0{,}3 \end{bmatrix} \tag{6}$$

Załóżmy, że stanami alarmowymi są stany: 1, 5, 6. Jeśli znajdujemy się w stanie 3, to $p_{31} = 0{,}0$; $p_{35} = 0{,}01$; $p_{36} = 0{,}0$; zatem będzie $\sum_{j=1,5,6} p_{3j} = 0{,}01$

Jeżeli uprzedzenie alarmu jest bardzo istotne dla procesu, to możemy założyć graniczne prawdopodobieństwo przejścia na poziomie $P_g = 0{,}01$ i wtedy sygnalizujemy możliwość alarmu ze stanu 3.

Przy niewielkiej wadze informacji o alarmie poziom ten ustalamy np. 0,2 lub 0,1.

Korzystając z pojęcia prawdopodobieństwa przejścia możemy określić zależności statystyczne nie tylko dla pomiarów sąsiadujących ze sobą. Np. prawdopodobieństwo znalezienia się w drugim pomiarze w stanie x_k, gdy w pierwszym byliśmy w x_i wynosi p_{ik}. Zdarzenie, że w trzecim pomiarze znajdziemy się teraz w stanie x_j, wymaga przejścia ze stanu x_i do stanu x_k oraz ze stanu x_k do stanu x_j. Zatem:

$$p_{ij(2)} = \sum_k p_{ik} p_{kj} \tag{7}$$

oznacza prawdopodobieństwo dojścia z x_i do x_j po każdej możliwej ze względu na x_k drodze. Wyższe prawdopodobieństwa $p_{ij(n)}$ dla n kroków można obliczać analogicznie.

Jeśli proces przebiega bardzo szybko, a przy jego obserwacji dysponujemy stosunkowo wolną maszyną, to można teraz określać stany przedalarmowe na dwa lub trzy pomiary naprzód.

Odpowiednie macierze przejść (M_2 i M_3) jest łatwo obliczyć z macierzy przejść w jednym kroku. Wystarczy zauważyć, że w macierzy M_2 każdy wyraz leżący na przecięciu i-tego wiersza i j-tej kolumny ma być sumą iloczynów i-tego wiersza macierzy M_1 przez wyrazy jej j-tej kolumny. Zgodnie z regułą mnożenia macierzy otrzymujemy:

$$M_2 = (M_1)^2 \tag{8}$$

oraz analogicznie:

$$M_3 = (M_1)^3 \tag{9}$$

4. Niejednorodne łańcuchy Markowa

Powyższe wywody odnoszą się do procesów stacjonarnych. Ale nawet jeśli uznamy, że obejmują one procesy zmieniające się bardzo wolno, to i tak omówiona metoda przewidywania alarmów znajduje tu niewielkie zastosowanie. Wystarczy, jeśli to konieczne, dokonać obliczeń raz i posługiwać się nimi stale w dalszym przebiegu procesu.

Lepiej można analizę tę wykorzystać do rozpatrzenia procesów o zmieniających się prawdopodobieństwach przejść. Ich dyskretne pomiary stanowić będą niejednorodny łańcuch Markowa.

Wymaga to stałej modyfikacji macierzy przejść M_1 (możemy założyć, że zmiany nie są gwałtowne i dokonywać modyfikacji nie w każdym kroku, lecz co pewien okres T).

Modyfikację zrealizujemy konstruując na podstawie przejść w ostatnim okresie T macierz modyfikującą M_T. Prawdopodobieństwa p_{ij} wchodzące w skład macierzy M_1 są ilorazem liczby przejść ze stanu x_i do stanu x_j przez liczbę wszystkich pomiarów, w których osiągnęliśmy stan x_i. Zatem modyfikację musimy przeprowadzić tak, aby w nowej zmodyfikowanej macierzy $M_{1\text{zmod}}$. prawdopodobieństwa przejść uwzględniały pomiary ostatniego okresu T:

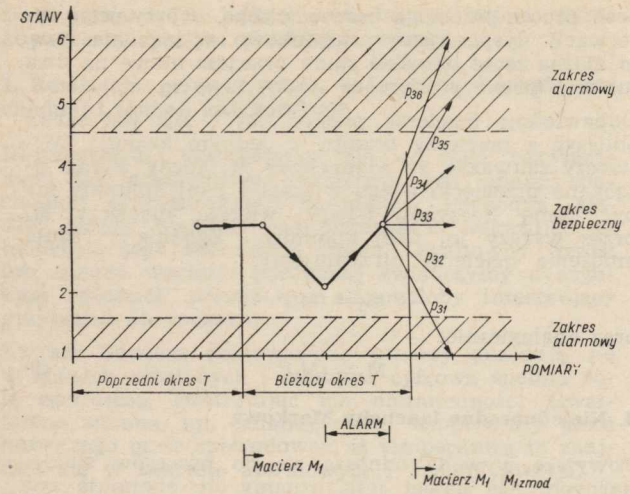
$$M_{1\,\text{zmod}} = \frac{M_1 N + M_T}{N + 1} \tag{10}$$

gdzie: $N + 1$ — to ilość dotychczasowych okresów T. Rozpatrzmy zasadę takiej modyfikacji, posługując się przykładem (6). Załóżmy, że prawa rządzące procesem w stanach bezpiecznych i alarmowych są jednakowe oraz że nie trzeba przyjmować żadnych warunków brzegowych (tzn. nie istnieją specjalne efekty na brzegach zakresu, jak np. „odbicie" lub „pochłanianie").

W pewnym kroku pomiarowym dysponujemy zatem macierzą przejść $M_{1(6\times6)}$. Należy tu zwrócić uwagę, iż założenie o konieczności odczekania pewnego czasu od początku procesu, które przyjęliśmy na początku punktu 3, nie jest konieczne. Zamiast bowiem określania prawdopodobieństwa przejść na podstawie historii procesu, możemy założyć rozkład tego prawdopodobieństwa w postaci najlepiej ujmującej zmiany danego przebiegu (np. rozkład normalny lub Poisson'a).

W pierwszym pomiarze znajdujemy się, powiedzmy, w stanie trzecim (byliśmy w nim także w poprzednim pomiarze). Obliczmy prawdopodobieństwo znalezienia się w zakresie alarmowym: $\sum_{j=1,5,6} p_{i=3j} = 0{,}01 < 0{,}1$ (przyjęty poziom) — zatem nie sygnalizujemy alarmu. W drugim pomiarze otrzymujemy np. stan drugi $\sum_{j=1,5,6} p_{i=2j} = 0{,}1$ — alarm jest już w tym przypadku sygnalizowany.

Maszyna, uruchamiając program alarmowy oddziałuje na proces tak, aby sprowadzić go do stanu właściwego.



Rys. 1

Dlatego też w trzecim pomiarze dostajemy znowu stan trzeci — sygnalizacja alarmowa ustaje (rys. 1).

Niech okres T obejmuje trzy pomiary; zatem z trzech pomiarów otrzymujemy nową, modyfikującą macierz $M_{T(6\times6)}$. Wiersze, w których byłyby same zera przepisujemy z macierzy M_1 (jeśli bowiem w ostatnim okresie T nie wychodziliśmy ze stanów odpowiadających tym wierszom, rozkład prawdopodobieństwa dla tych stanów nie może ulec zmianie w $M_{1\,zmod}$).

$$M_T = \begin{bmatrix} 0{,}3 & 0{,}4 & 0{,}2 & 0{,}08 & 0{,}02 & 0{,}0 \\ 0{,}0 & 0{,}0 & 1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}5 & 0{,}5 & 0{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}3 & 0{,}4 & 0{,}25 & 0{,}05 \\ 0{,}02 & 0{,}08 & 0{,}1 & 0{,}3 & 0{,}3 & 0{,}2 \\ 0{,}0 & 0{,}02 & 0{,}08 & 0{,}2 & 0{,}4 & 0{,}3 \end{bmatrix} \qquad (11)$$

Musimy więc dotychczas używaną macierz M_1 zmodyfikować tak, aby uwzględniała ona wkład ostatniego okresu historii procesu. Dla naszego przykładu załóżmy, że $N = 1$.

$$M_{1\,zmod} = \begin{bmatrix} 0{,}3 & 0{,}4 & 0{,}2 & 0{,}08 & 0{,}02 & 0{,}0 \\ 0{,}025 & 0{,}125 & 0{,}75 & 0{,}075 & 0{,}025 & 0{,}0 \\ 0{,}0 & 0{,}255 & 0{,}495 & 0{,}245 & 0{,}005 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}3 & 0{,}4 & 0{,}25 & 0{,}05 \\ 0{,}02 & 0{,}08 & 0{,}1 & 0{,}3 & 0{,}3 & 0{,}2 \\ 0{,}0 & 0{,}02 & 0{,}08 & 0{,}2 & 0{,}4 & 0{,}3 \end{bmatrix} \qquad (12)$$

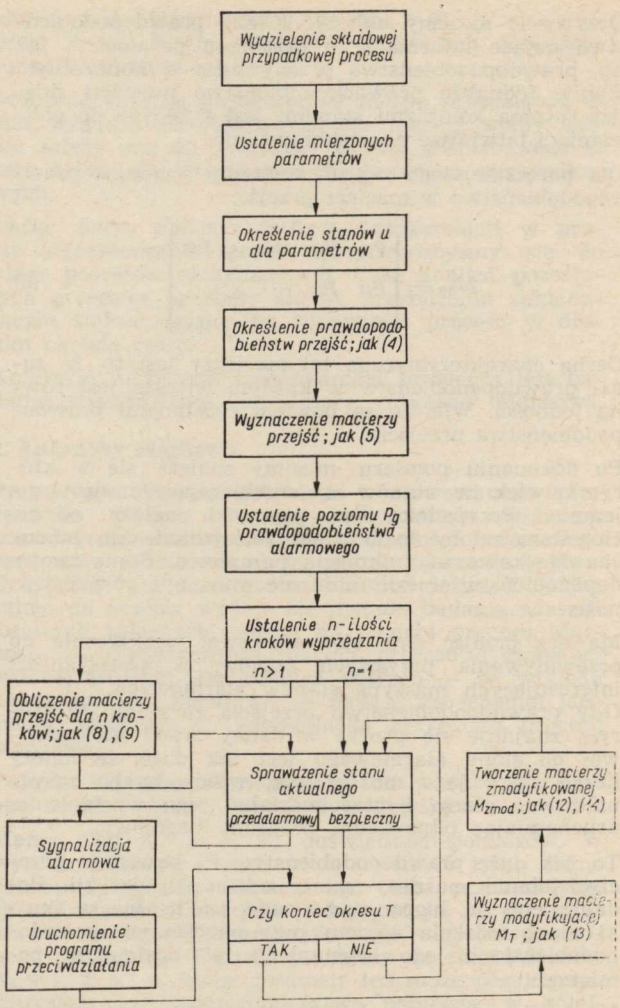
Nowa macierz $M_{1\,zmod}$ będzie służyć jako M_1 w trzech następnych pomiarach.

Maszyna może na początku każdego okresu pomiarowego obliczać dla wszystkich stanów dozwolonych prawdopodobieństwa przejścia do stanów alarmowych. Następnie na tej podstawie kwalifikuje niektóre z nich jako stany przedalarmowe. Tak będzie, gdy jest mało stanów i modyfikacja następuje po wielu pomiarach. Natomiast, gdy jest dużo stanów i zaledwie parę pomiarów w okresie, ta metoda nie jest korzystna. Lepiej wówczas liczyć prawdopodobieństwo przejścia do stanów alarmowych dla każdego pomiaru.

Duża liczba pomiarów i modyfikacji macierzy może prowadzić do zatarcia się, rozmycia charakterystyki prawdopodobieństw. Dzieje się tak zwłaszcza wówczas, gdy tendencje zmian parametru same ulegają częstym zmianom. Dlatego też będziemy rozważać przy modyfikacji jedynie macierze pewnej liczby ostatnich pomiarów. Będzie to jakby statystyczne śledzenie dynamiki procesu.

5. Możliwość stosowania łańcuchów Markowa

Można więc już w tym miejscu przedstawić algorytm realizujący wyprzedzanie procesu (rys. 2).



Rys. 2

Pozostaje jeszcze do rozważenia problem, czy każdy z obserwowanych procesów może być uznany za proces Markowa i analizowany w powyższy sposób. Oczywiście tak nie jest.

Mamy np. zbiornik z dwiema, nie reagującymi ze sobą cieczami A i B. Chcemy powiększyć w tym zbiorniku przewagę jednej z nich (obojętnie której). Przyrząd pomiarowy pobiera więc małą próbkę z przypadkowego miejsca w zbiorniku i rejestruje jej zawartość jako stan z przewagą cieczy A lub cieczy B. Następnie dolewa r litrów tej cieczy, której było więcej w próbce. Na początku były zatem następujące prawdopodobieństwa obu stanów pomiarowych:

$$p_A = \frac{a}{c}; \quad p_B = \frac{b}{c} = 1 - p_A, \quad \text{gdzie } a \text{ — liczba litrów}$$

cieczy A, b — liczba litrów cieczy B, c — pojemność zbiornika w litrach ($c = a+b$). Prawdopodobieństwo otrzymania w drugiej próbce stanu A (z przewagą cieczy A), gdy w pierwszym pomiarze otrzymano także stan A, wynosi $p_{AA} = \frac{a+r}{c+r}$.

Dalej w trzecim pomiarze $p_{AAA} = \frac{a+2r}{c+2r}$.

Natomiast „dojście po innej drodze" (tzn., gdy w pierwszym pomiarze otrzymano stan B) do stanu A w trzecim pomiarze ma prawdopodobieństwo

$$p_{BAA} = \frac{a+r}{c+2r}, \quad \text{ponieważ } p_{BA} = \frac{a}{c+r}.$$

Zatem prawdopodobieństwo przejścia ze stanu A w drugiej próbie do stanu A w trzeciej jest zależne od

pomiarów w pierwszej próbie. Nie jest to więc łańcuch Markowa.

Uznajmy jednak za stany nie zawartości próbek, lecz liczbę pomiarów, w których otrzymano przewagę cieczy A. Tę liczbę oznaczymy np. i w dotychczasowej liczbie n wszystkich pomiarów (tzn. 0 i n). Zatem po n pomiarach znajdujemy się w stanie x_i. W następnym $n+1$-szym pomiarze możemy przejść do x_{i+1} lub pozostać w stanie x_i. Prawdopodobieństwa przejścia do pozostałych stanów wynoszą 0. Wszystkie prawdopodobieństwa przejścia nie zależą od stanów poprzednich. Tak opisany proces stał się łańcuchem Markowa.

Trudno ocenić stopień ogólności operacji opisanej w tym przykładzie, jednak może ona znacznie zwiększyć zakres stosowalności omawianej metody.

Podany algorytm dla $M_{(40\times40)}$ ma zostać użyty w roku 1972 do nadzorowania pracy elektrowni w Milazzo na Sycylii; T obejmuje 10 pomiarów, proces jest 16-parametrowy. Były też podejmowane próby zastosowania go w cementowni we Francji.

BIBLIOGRAFIA

[1] Hellwig — „Elementy rachunku prawdopodobieństwa i statystyki matematycznej", PWN 1965 Warszawa.

[2] Dynikin, Juszkiewicz — „Twierdzenia i problemy procesów Markowa", PWN, Warszawa 1970.

[3] Rosenblatt — „Procesy stochastyczne", PWN Warszawa 1967.

[4] Sarymsakow — „Osnowy tieorii procesów Markowa", Moskwa 1954.

[5] Mesarovic — „Self — organising control systems", „IEE Transactions Apl. and Industry", 1964, nr 74.

RYSZARD FRYDRYCHOWSKI

Instytut Energetyki
Warszawa

65.012.122

Programowanie liniowe w liczbach całkowitych

Artykuł stanowi wprowadzenie w teorię programowania liniowego w liczbach całkowitych. Naszkicowano ideę podstawowych algorytmów pochodzących od Dantziga, Gomory'ego i Landa-Doig.
Autor zwraca uwagę na szerokie możliwości wykorzystania programowania liniowego całkowitoliczbowego do optymalizacji zagadnień gospodarki narodowej i wskazuje potrzebę praktycznej realizacji tych algorytmów na zainstalowanych w kraju maszynach cyfrowych.

Spośród wielu metod badań operacyjnych, najwięcej zastosowań w Polsce znalazło programowanie liniowe. Ma w tym udział zarówno mnogość procesów liniowych występujących w zagadnieniach ekonomicznych i technicznych, jak również łatwość modelowania tych procesów w języku programowania liniowego. Uwzględniając ponadto dobre w tej dziedzinie oprogramowanie zainstalowanych w kraju maszyn cyfrowych, otrzymujemy pełniejszy obraz czynników, które zdecydowały, że programowanie liniowe jest dziś silnym narzędziem wykorzystywanym z pożytkiem dla gospodarki narodowej przez rzesze ekonomistów, inżynierów i matematyków.

Zwyczajne zagadnienia z programowania liniowego przeważnie formułuje się następująco:

znaleźć wektor X, który minimalizuje funkcję liniową

$$z = C \cdot X \tag{1}$$

przy warunkach:

$$A \cdot X = D \tag{2}$$
$$X \geqslant 0$$

gdzie C — wektor o wymiarze wektora X (np. n), A — macierz $(m \times n,\ m < n)$, D — wektor m wymiarowy, wszystko o elementach rzeczywistych.

Punkt optymalny, tzn. spełniający (2), który czyni formę (1) minimalną, leży zawsze na brzegu wypukłego wielościanu (2). To podstawowe twierdzenie dla algorytmu simplex Dantziga nie daje się przenieść niestety na programowanie liniowe z dodatkowym warunkiem poszukiwania rozwiązania całkowitoliczbowego.

Niewiele znaczące mogłoby się zdawać na pierwszy rzut oka dodatkowe wymaganie, aby współrzędne wektora X były liczbami całkowitymi, kwalifikuje nowy problem programowania liniowego do zupełnie innej klasy zagadnień, a mianowicie do programowania niewypukłego. Stąd pochodzą trudności, jakie napotykano od kilkunastu lat, przy poszukiwaniu efektywnych algorytmów rozwiązania problemu programowania liniowego przy zmiennych całkowitych.

Znalezienie w ostatnich latach takich algorytmów skłoniło M. Simonnarda, autora znanej książki o programowaniu liniowym [8], do uznania tych wyników za najważniejsze i najbardziej obiecujące, jakie uzyskano w teorii programowania liniowego w ostatnim czasie.

Programowanie liniowe przy zmiennych całkowitych stwarza bowiem możliwości rozwiązywania wielu zagadnień programowania nieliniowego, niewypukłego, a więc zagadnień, których nie można rozwiązywać klasycznym aparatem, choćby z powodu braku wypukłości zbioru rozwiązań dopuszczalnych czy funkcji celu.

Zadaniem niniejszego artykułu jest szkicowe przedstawienie idei podstawowych rozwiązań, uzyskanych w ciągu ostatnich lat, w zakresie programowania liniowego przy zmiennych całkowitych.

Sformułowanie, klasyfikacja i geometryczna interpretacja zagadnienia

Sformułowanie problemu liniowego całkowitoliczbowego niewiele odbiega od zwykłej postaci (1), (2). Oprócz podanych tam wymagań odnośnie do macierzy A i wektorów C, D, X dodaje się jeszcze jeden warunek, aby współrzędne wektora — rozwiązania X były całkowite dla pewnego zbioru indeksów J:

$$x_j \text{ całkowite dla } j \in J \tag{3}$$

przy czym, jeśli wprowadzić oznaczenie zbioru wszystkich indeksów $N = \{1, 2, \ldots, n\}$, to $J \in N$.

W zależności od wzajemnych relacji między zbiorami J i N mogą być rozpatrywane trzy zagadnienia:

1. Jeżeli zbiór J jest pusty, mamy zwyczajne programowanie liniowe.

2. Zbiór $J = N$, mamy wtedy tzw. *„all integer"*, czyli nazywane w języku polskim zagadnienie całkowite w liczbach całkowitych, albo krótko zagadnienie całkowite.

3. $N - J \neq 0$, otrzymujemy tzw. *„mixed integer"*, czyli problem mieszany, w języku polskim nazywany problemem częściowo całkowitym.

Rozwiązanie zadania grupy 2) sprowadza się do znalezienia punktu kratowego należącego do zbioru Z wyznaczonego przez warunki (2). Punkt ten ma minimalizować funkcję (1).

Pojęcie punktu kratowego, tzn. punktu przestrzeni o wszystkich współrzędnych całkowitych, pochodzi z działu matematyki — teoria liczb, w szczególności z tzw. geometrycznej teorii liczb. Punkt ten odgrywa dużą rolę w oszacowaniach całek i w teorii grup.

Interpretacja geometryczna problemu całkowitoliczbowego jest dość oczywista. Rozwiązanie optymalne stanowi punkt kratowy, który należy do Z i jednocześnie znajduje się najbliżej hiperpłaszczyzny wyznaczonej przez funkcję celu przechodzącą przez punkt optymalny dla problemu liniowego bez warunku (3).

Jeżeli punkt optymalny dla problemu w zwykłym sensie jest jednocześnie punktem kratowym, to rozwiązanie jest natychmiastowe. Na ogół jednak tak nie będzie. Najczęściej pierwszy z brzegu punkt kratowy będzie leżał wewnątrz obszaru Z.

Można wówczas wykrywać go przez badanie odległości od hiperpłaszczyzny z_0, ewentualnie przez ścinanie zbioru Z odpowiednimi hiperpłaszczyznami, gwarantującymi, że w odciętej części zbioru Z nie znajdzie się żaden punkt kratowy. Postępując w ten sposób hiperpłaszczyzna z_0 przybliża się coraz bardziej do pierwszego z brzegu punktu kratowego, znajdującego się w zbiorze Z. Odpowiedni dobór hiperpłaszczyzn tnących (tzw. cięć) stanowi istotę większości algorytmów rozwiązywania problemów liniowych całkowitoliczbowych.

Forma cięć w algorytmie Dantziga

Pierwszą pracę, dającą konkretną sugestię w kierunku rozwiązania zagadnienia programowania całkowitoliczbowego, opublikował G. B. Dantzig już w roku 1959 [1]. W pracy tej Dantzig skonstruował ciąg hiperpłaszczyzn tnących (cięć), które obcinają zbiór rozwiązań dopuszczalnych Z, powodują przybliżanie punktów kratowych wewnętrznych do brzegu zmodyfikowanego obszaru.

Jak wiadomo, rozwiązanie dopuszczalne w zwykłym problemie liniowym otrzymuje się z równości:

$$X_B = Y_0 - \sum_{j \in R} Y_j \cdot x_j \tag{4}$$

otrzymanej z wyrażenia m zmiennych x_j przez pozostałe $n - m$ zmiennych niezależnych w taki sposób, aby składowe wektora — wyrazu wolnego Y_O były dodatnie. Zmienne zależne x_j nazywa się też często bazowymi, stąd oznaczenie wektora złożonego z tych m zmiennych x_j przez X_B. Pozostałe zmienne niezależne nazywa się zmiennymi wtórnymi, a ich indeksy j tworzą zbiór R. Po przyjęciu liczbowych wartości zmiennych wtórnych $x_j = 0$ dla $j \in R$, otrzymuje się rozwiązanie dopuszczalne będące zarazem punktem wierzchołkowym zbioru Z.

Podstawiając (4) do funkcji celu (1) otrzymuje się

$$z(X_R) = z_0 - \sum_{j \in R} (z_j - c_j)\, x_j \tag{5}$$

gdzie

$z_0 = C_B Y_O$

Jeśli w (5) wszystkie różnice spełniają nierówność $z_j - c_j \leqslant 0$, to rozwiązanie dopuszczalne $X_B = Y_O$ jest rozwiązaniem optymalnym zwykłego programu liniowego. Gdyby rozwiązanie to było jednocześnie całkowite, wówczas byłoby również rozwiązaniem optymalnym problemu całkowitoliczbowego. Najczęściej jednak tak nie będzie. Niech np. jedna ze współrzędnych y_{oi} wektora Y_O nie będzie całkowita, wtedy aby uczynić zadość całkowitości odpowiedniej współrzędnej x_{Bi} z wektora X_B musi być też we wzorze (4) choć jedno $x_j > 0$ dla $j \in R$. A ponieważ $x_j \in$ ma być całkowite, więc na pewno musi zachodzić nierówność:

$$\sum_{j \in R} x_j \geqslant 1 \tag{6}$$

Hiperpłaszczyzna $\sum_{j \in R} x_j = 1$ jest cięciem w postępowaniu Dantziga. Warunek (6) zostaje dodany do systemu warunków (2) i ponownie rozwiązuje się zwykłe zadanie programu liniowego.

Proces dodawania warunków typu (6) kontynuuje się tak długo, aż osiągnie się pierwsze rozwiązanie całkowitoliczbowe, które jest rozwiązaniem optymalnym rozszerzonego problemu.

Algorytm Dantziga był wypróbowany w pewnej liczbie przypadków i zbieżność otrzymywano zawsze. Niemniej, jak dowodzą R. Gomory i A. Hoffman [2] mogą istnieć takie przypadki, w których procedura Dantziga nie doprowadzi do rozwiązania optymalnego. W innych pracach [3, 4] w latach 1958—60 R. Gomory podał systematyczny i bardziej rozwinięty algorytm otrzymywania cięć, które prowadzą do znalezienia optymalnego rozwiązania problemu (1) — (3) w skończonej liczbie kroków.

Algorytm „all integer" R. Gomory'ego

Podobnie jak u Dantziga, tak i tu pierwszym krokiem w rozwiązaniu zagadnienia (1) — (3) będzie rozwiązanie go z pominięciem warunku (3).

Rozwiązaniem optymalnym będzie $X_B = Y_O$ otrzymane ze wzoru (4) przez podstawienie $x_j = 0$ dla $j \in R$ przy każdym $z_j - c_j \leqslant 0$. Gdyby wszystkie składowe wektora Y_O były całkowite, wówczas byłoby to również rozwiązanie rozszerzonego problemu (1) — (3). Na ogół jednak tak nie będzie. Przypuśćmy więc, że choć jedna ze składowych wektora Y_O, np. y_{lo} nie jest całkowita. Równanie l-te w układzie (4) jest postaci

$$y_{Bl} = y_{lo} - \sum_{j \in R} y_{lj}\, x_j \tag{7}$$

Zarówno składową y_{lo} jak i y_{lj} można rozłożyć na sumę części całkowitej i ułamkowej, tzn.

$$y_{lo} = [y_{lo}] + \{y_{lo}\};\quad y_{lj} = [y_{lj}] + \{y_{lj}\}\ \ j \in R$$

gdzie

$$0 \leqslant \{y_{lj}\} < 1$$

oraz

$$0 < \{y_{lo}\} < 1 \tag{8}$$

ponieważ y_{lo} nie jest całkowite.
Podstawiając podane rozkłady do (7) otrzymuje się

$$x_{Bl} = [y_{lo}] - \sum_{j \in R} [y_{lj}]\, x_j + \{y_{lo}\} - \sum_{j \in R} \{y_{lj}\}\, x_j$$

czyli

$$\{y_{lo}\} - \sum_{j \in R} \{y_{lj}\}\, x_j$$

jest liczbą całkowitą. Jest również liczbą niedodatnią, gdyż

$\sum_{j \in R} \{y_{ij}\} x_j \geqslant 0$. a $\{y_{lo}\}$ spełnia (8).

Otrzymujemy więc nierówność

$$\sum_{j \in R} \{-y_{lj}\} x_j \leqslant -\{y_{lo}\} \tag{9}$$

Jest to dodatkowy warunek — cięcie, którym uzupełnia się układ nierówności 2 i rozwiązuje ponownie zwykły program liniowy. Do rozwiązania rozszerzonego zadania stosuje się algorytm dualny, w którym jak wiadomo dopuszcza się „ujemność" niektórych zmiennych. Jest to konieczne, ponieważ po wprowadzeniu zmiennej bilansującej s_1 do (9) dostaje się

$$s_1 = -\{y_{lo}\} + \sum_{j \in R} \{y_{lj}\} x_j \tag{10}$$

Równość ta dopisana do układu (2) powoduje, że w rozwiązaniu optymalnym pojawi się składowa — $\{y_{lo}\}$, która jest ujemna na mocy (8) przy jednoczesnym spełnieniu $z_j - c_j \leqslant 0$ dla wszystkich $j \in R$.

Jest to rozwiązanie niedopuszczalne dla zagadnienia pierwotnego i dlatego odwołujemy się do algorytmu dualnego, który tego nie wymaga.

Udowodniono, że procedura Gomory'ego jest zbieżna przy dodatkowych warunkach ograniczoności zmiennych x_j oraz całkowitości elementów macierzy A. Spełnienie tych wymagań w praktycznych obliczeniach można zawsze osiągnąć, gdyż ograniczoność zmiennych determinuje poprawność postawionego zadania praktycznego, zaś całkowitość elementów a_{ij} uzyskuje się przez dobór odpowiedniej skali liczbowej. Przedstawione szkicowo dwa algorytmy znajdowania rozwiązań optymalnych całkowitych w programowaniu liniowym stanowią podstawę metod w grupie (2). Jeśli chodzi natomiast o zagadnienia mieszane, to znalezione tam algorytmy są znacznie mniej efektywne. Istniejący algorytm podany również przez R. Gomory'ego w roku 1960 [5] posłużył już do rozwiązania wielu programów mieszanych, niemniej jak stwierdza G. Hadley [6] algorytm ten nie jest jeszcze wszechstronnie zbadany.

Innym podejściem do znalezienia rozwiązania problemu liniowego, całkowitoliczbowego, stanowiącego szczególny przypadek programu dyskretnego, charakteryzuje się praca dwóch matematyków angielskich A. H. Landa i A. G. Doig [7]. Opublikowali oni w roku 1960 zupełnie odmienny algorytm niż metody cięć. Jest to algorytm nieco pracochłonniejszy, ale też bardziej intuicyjnie uchwytny, łatwy do zaprogramowania i wspólny dla obu grup zagadnień całkowitego i mieszanego.

Algorytm kombinatoryczny Landa — Doig

W odróżnieniu od postępowań Dantziga czy Gomory'ego, w których rozwiązywało się coraz bardziej złożone programy liniowe, procedura L-D polega na rozpatrywaniu ciągu problemów liniowych, w których liczba zmiennych systematycznie maleje. Jest to jedna z zalet metody, niemniej należy stwierdzić, że w ogólności na znalezienie rozwiązania optymalnego tą metodą trzeba wykonać więcej kroków niż np. metodą Gomory'ego. Uwaga ta dotyczy szczególnie rozwiązywania zadań o dużej liczbie zmiennych. Procedurę L-D bardziej poglądowo ilustruje się przy pomocy drzewa. Jako wierzchołek tego drzewa (punkt początkowy) przyjmuje się rozwiązanie optymalne zadania (1), (2). Rozwiązaniu w punkcie początkowym odpowiada wartość funkcji celu z_o. Z tego punktu wyprowadza się gałęzie odpowiadające nadaniu pierwszej zmiennej x_j dwóch wartości całkowitych leżących najbliżej wartości x_j^0 z rozwiązania początkowego. Najbliższymi liczbami całkowitymi będą oczywiście $[x_j^0]$ i $[x_j^0] + 1$. Jako numer j przyjmuje się np. najmniejszy indeks zmiennej, która ma być całkowita $j \in J$, a w rozwiązaniu początkowym nie osiągnęła wartości całkowitej, $\{x_j^0\} > 0$. Przyjęcie konkretnych wartości na zmienną zmniejsza wymiar zadania, które rozwiązuje się znów np. algorytmem simplex. W wyniku tego otrzymuje się nowe węzły kończące nowo utworzone gałęzie. Węzły, w których nie otrzymano rozwiązania, albo funkcja celu osiągnęła mniejszą wartość niż w kroku poprzednim odrzuca się z dalszej analizy. Po ustaleniu jednej zmiennej ustala się następne wykonując dla nich rozgałęzienia i węzły.

Wartości całkowite przyjmuje się oczywiście tylko dla tych zmiennych, które zostały wyszczególnione w warunku (3). Końcowym rozwiązaniem jest to rozwiązanie, które dając najmniejszą wartość funkcji z czyni zadość całkowitości wszystkich współrzędnych, o numerach należących do J. Poszukiwane rozwiązanie tworzy ciąg gałęzi o ustalonych całkowitych wartościach zmiennych x_j dla $j \in J$, po których można dojść do węzła z_k najmniej odległego od wierzchołka drzewa z_o. Przy czym za węzeł znajdujący się najbliżej początkowego z_o uważa się ten węzeł z_k, dla którego różnica wartości funkcji celu $z_k - z_o$ = minimum. Pełne rozwiązanie wymaga wykreślenia z każdego węzła nie tylko dwóch gałęzi, ale tylu na ile pozwala wielkość zbioru Z, tzn. jakie są zakresy poszczególnych zmiennych x_j aby nie wykraczać poza zbiór Z.

Algorytm L-D jest bardzo wygodny szczególnie w stosowaniu do tzw. programowania zero-jedynkowego (pseudoboolowskiego). Programowanie zero-jedynkowe jest utworzone przez warunki (1) — (3) uzupełnione dodatkowym ograniczeniem

$$X \leqslant I \tag{11}$$

gdzie I jest wektorem złożonym z samych jedynek. W konsekwencji tego faktu każda współrzędna może przyjmować co najwyżej jedną z dwóch liczb 0 albo 1, a więc z każdego węzła drzewa mogą wychodzić co najwyżej dwa rozgałęzienia.

Programowanie zero-jedynkowe ma szerokie zastosowanie we wszelkiego rodzaju zagadnieniach decyzyjnych. Przyjęcie bowiem definicji zmiennych $x_j = 1$ — jako realizacja pewnego przedsięwzięcia, lub jakiegoś etapu tego przedsięwzięcia:

$x_j = 0$ — rezygnacja z realizacji przedsięwzięcia, pozwala niemal w natychmiastowy sposób zapisać zbiór warunków ograniczających i funkcję celu.

Programowanie zero-jedynkowe ma ponadto zastosowania do aproksymacji zagadnień nieliniowych. Znane jest twierdzenie, które powiada, że każde zagadnienie programowania nieliniowego można metodą aproksymacyjną sprowadzić do programowania liniowego zero-jedynkowego. Jest to również skuteczna metoda do rozwiązywania zagadnień z ograniczeniami alternatywnymi, czyli zagadnień o niespójnych zbiorach rozwiązań dopuszczalnych. Kilka przykładów zastosowania tego typu programowania można znaleźć w [8].

Wnioski

Podane szkicowo trzy algorytmy rozwiązywania zagadnień programowania liniowego na zmiennych całkowitych oczywiście nie wyczerpują wszystkich algorytmów.

Znanych jest jeszcze kilka procedur, o których nie wspomniano z powodu braku miejsca.

Celem tej publikacji nie było bowiem podanie szczegółowych algorytmów dających się np. bezpośrednio przenieść na program dla maszyny cyfrowej, ponieważ opisy tych algorytmów można znaleźć w pracach źródłowych. Intencją autora było jednak podanie w miarę przystępnego wprowadzenia w teorię programowania całkowitoliczbowego, przedstawienie głównych kierunków badań i zarysowanie przyjętych w tym przedmiocie metod.

Programowanie całkowitoliczbowe jest interesującą dziedziną badań operacyjnych zarówno pod względem teoretycznym, jak też pod względem możliwości przeróżnych zastosowań w gospodarce narodowej.

Można podać wiele modeli liniowych całkowitoliczbowych, choćby np. w zastosowaniu do zagadnień przemysłu energetycznego. Problem ten ze względu na swe rozmiary będzie przedmiotem innej publikacji. Podsumowując przytoczone tu informacje warto zwrócić uwagę na istniejące już dziś w Polsce możliwości praktycznej realizacji algorytmów całkowitoliczbowych, nawet przy obecnym stanie zainstalowanych w kraju maszyn cyfrowych.

BIBLIOGRAFIA

[1] G. B. Dantzig: Note on solving linear programs in integers. Naval Research Logistics Quarterly, 6, 1959.

[2] R. E. Gomory, A. J. Hoffman: On the convergense of integer programming process. Naval Research Logistics Quarterly, 10, 1963.

[3] R. E. Gomory: Outline of an algorithm for integer solutions to linear programs. Bulletin of the American Mathematical Society, 64, 1958.

[4] R. E. Gomory: All integer programming algorithm. IBM Research Center, Research Report RC-189, 1960.

[5] R. E. Gomory: An algorithm for the mixed integer problem, P-1885, The RAND Corp., Juny 1960.

[6] G. Hadley: Nonlinear and dynamic programming, Addison-Wesley Publishing Comp., 1964.

[7] A. H. Land, A. G. Doig: An automatic method of solving discrete programming problems, Econometrica, 28, 1960.

[8] M. Simonnard: Programmation linéaire, Dunod, Paris 1962. Tłumaczenie polskie PWN, Warszawa, 1967.

PIOTR PERKOWSKI

Instytut Maszyn Matematycznych
Warszawa

681.322.06

ZAM-GPSS – język do symulacji procesów dyskretnych

Przedstawiono język do symulacji procesów dyskretnych ZAM—GPSS, zrealizowany na EMC ZAM-41 w Instytucie Maszyn Matematycznych w Warszawie. Omówiono elementy i budowę języka oraz podano przykład programu.

1. Wstęp

W ciągu ostatnich dziesięciu lat byliśmy świadkami bardzo szybkiego rozwoju zastosowań maszyn cyfrowych do symulacji procesów dyskretnych. Specjalizowane języki symulacyjne stanowią dziś szeroko wykorzystywane narzędzie w badaniach złożonych systemów w wielu dziedzinach nauki i techniki, takich jak gospodarka materiałowa, systemy łączności, systemy masowej obsługi, zarządzanie itp. Doświadczenie wskazuje, że programowanie problemów symulacyjnych w językach algorytmicznych typu FORTRAN czy ALGOL jest na ogół bardzo uciążliwe i pracochłonne. Konieczność wyrażenia pojęć specyficznych dla symulacji w językach tego typu znacznie utrudnia i komplikuje proces budowy i uruchamiania modelu.

Celem niniejszego artykułu jest zapoznanie grona specjalistów zainteresowanych wykorzystaniem maszyn cyfrowych do celów symulacji procesów dyskretnych z językiem symulacyjnym ZAM-GPSS zrealizowanym w IMM na maszynie cyfrowej ZAM-41. Język ten jest odpowiednikiem języka GPSS/360 (General Purpose System Simulator). Wydaje się, że wykorzystanie efektywnych metod symulacyjnych w praktyce badawczej i inżynierskiej jest w Polsce wciąż jeszcze mało rozpowszechnione.

Praktyczne wykorzystanie języka ZAM-GPSS powinno przyczynić się do szerszego spopularyzowania nowoczesnych technik symulacyjnych, posłużyć do zebrania doświadczeń eksploatacyjnych, które umożliwią szybszy rozwój tej dziedziny zastosowań m.c. w kraju.

2. Charakterystyka języka symulacyjnego ZAM-GPSS

ZAM-GPSS jest językiem symulacyjnym o strukturze blokowej. Blokowa struktura modelu jest cechą różniącą GPSS od innych istniejących języków wyspecjalizowanych do symulacji procesów dyskretnych. Opis blokowy systemu jest szczególnie przejrzysty, łatwy w interpretacji i powszechnie stosowany przez analityków. Funkcje wypełniane przez poszczególne bloki (np. przerwania, kolejki) są bliskie realiom modelowanego procesu, co znacznie ułatwia programowanie. Budowa modelu systemu, przejście od schematu czynnościowego do programu symulacyjnego są, w tej sytuacji, bardzo uproszczone.

Do podstawowych cech języka ZAM-GPSS można zaliczyć:

1. Rozbudowany aparat diagnostyczny i uruchomieniowy.

2. Rozbudowany system automatycznego zbierania informacji statystycznych o przebiegu symulacji. Informacje te są dostępne zarówno w czasie trwania procesu symulacji (np. na żądanie operatora), jak i po jego zakończeniu.

3. Możliwość graficznego przedstawienia wyników pośrednich lub końcowych (w postaci histogramu).

4. Możliwość dokonywania dowolnych zmian w modelu (zmiany struktury lub parametrów) w trakcie procesu symulacji — współpraca operatora z programem symulacyjnym, na bieżąco.

5. Możliwość przekazywania danych między niezależnymi programami symulacyjnymi.

6. Możliwość tworzenia biblioteki programów symulacyjnych (na taśmach magnetycznych).

Ze względu na parametry techniczne m.c. ZAM-41 realizacja języka ZAM-GPSS różni się od wersji oryginalnej i stanowi jedynie adaptację koncepcji języka GPSS/360.

3. Elementy i budowa języka

Model systemu (procesu) zaprogramowany w języku ZAM-GPSS może być rozpatrywany jako zbiór wzajemnie powiązanych operatorów logicznych i arytmetycznych reprezentujących wybrane aspekty modelo-

System	Wielkości aktywne	Inne wielkości	Atrybut
Ruch uliczny	Pojazdy	Przelotowość ulicy, czynniki regulujące ruch pojazdów	Prędkości i odległości między pojazdami
Taśma montażowa	Podzespoły, elementy	Urządzenia, robotnicy	Pracochłonność montażu poszczególnych podzespołów, wykorzystanie urządzeń
Linia transmisyjna	Meldunki	Pojemność kanału, urządzenie nadawcze i odbiorcze	Priorytety meldunków, długości meldunków
Bank	Klienci	Obsługa, możliwości dokonywania wypłat, stan wypłat	Bilans, kredyty
Sklep	Klienci	Towary, obsługa	Lista zakupów

wanego systemu. W każdym systemie (procesie) można wyróżnić grupę elementarnych wielkości lub pojęć abstrakcyjnych, za pomocą których można skonstruować opis tego systemu. Dla celów symulacji procesów dyskretnych w języku GPSS zdefiniowano 13 typów wielkości oraz wprowadzono określone założenia i konwencje formalne, odpowiadające podstawowym czynnościom lub warunkom w rzeczywistym systemie. Z każdą z podstawowych wielkości związane są zbiory atrybutów numerycznych i logicznych opisujących własności i stan tych wielkości w każdej chwili czasowej. W tabeli 1 zebrano przykłady systemów dyskretnych.

Proces symulacji można rozpatrywać jako sekwencję wzajemnie powiązanych, oddziałujących na siebie zdarzeń.

Zdarzeniem jest każda zmiana stanu modelu następująca pod wpływem wzajemnych oddziaływań między wielkościami.

Stan modelu jest całkowicie określony przez zbiór wartości numerycznych i logicznych wszystkich atrybutów, parametrów i wskaźników zdefiniowanych w języku GPSS, w wyróżnionej chwili czasowej.

W języku GPSS zdefiniowano następujące typy wielkości:

1. Bloki
2. Zadania
3. Urządzenia
4. Magazyny
5. Przełączniki logiczne
6. Zmienne arytmetyczne
7. Zmienne boolowskie
8. Funkcje arytmetyczne
9. Funkcje boolowskie
10. Kolejki
11. Tabele
12. Pamięci
13. Zbiory użytkownika.

Jedynymi wielkościami czynnymi, dynamicznymi są zadania. Operatorami bezpośrednio wpływającymi na stan modelu są bloki. Tak więc czynnikami stymulującymi zdarzenia są zadania — charakter tych zdarzeń zależy od aktualnego stanu modelu (a więc i od wszystkich poprzednich zdarzeń) oraz od rodzaju bloku (operatora). Przechodzenie zadań z bloku do bloku według schematu (modelu) ustalonego przez programistę odzwierciedla przebieg rzeczywistego procesu, pozwala aktualizować informacje statystyczne dotyczące każdej ze zdefiniowanych wielkości. Kierunki przepływu zadań w modelu, a więc decyzje dotyczące wyboru następnego bloku mogą być:
— zdeterminowane, tzn. narzucone przez strukturę programu
— losowe (probabilistyczne)
— logiczne, podejmowane na podstawie zdefiniowanych zależności logicznych, relacji pomiędzy standardowymi atrybutami numerycznymi, zależne od aktualnego stanu modelu.

Wejście zadania do bloku powoduje wykonanie określonej operacji, czyli zmianę stanu modelu (zdarzenie) polegającą na:
— wygenerowaniu nowych zadań
— zmianie standardowych atrybutów numerycznych lub logicznych jednej lub kilku wielkości.

W szczególności zmiana ta może dotyczyć atrybutów samego zadania, atrybutów innych wielkości, np. pamięci, magazynów, kolejek itp.
— niszczeniu zadań.

Po wykonaniu operacji związanych z danym blokiem zadanie usiłuje przejść do następnego bloku. Jednakże w przypadku, gdy zdarzenie związane z następnym blokiem nie może wystąpić, tzn. gdy aktualny stan modelu uniemożliwia wykonanie operacji, zadanie ulega wstrzymaniu do chwili usunięcia czynników blokujących jego przepływ. Zadanie to może np. reprezentować samochód oczekujący na zmianę świateł sygnalizacyjnych na skrzyżowaniu ulic lub informację oczekującą w buforze na zwolnienie kanału przesyłania.

Zadania wstrzymane w modelu w oczekiwaniu na spełnienie określonego warunku przejścia do następnego bloku ulegają oczywiście opóźnieniu. Do pomiarów ilościowych charakterystyk statystycznych tych opóźnień służą bloki związane z wielkością zwaną KOLEJKĄ.

Atrybutami numerycznymi charakteryzującymi aktualny stan kolejki są między innymi: średni stan pobytu zadania w kolejce, maksymalna ilość zadań w kolejce, średnia ilość zadań itp.

Zadania są nośnikami informacji reprezentującymi czynniki zewnętrzne względem modelowego systemu, które oddziałując na ten system w określonym czasie i miejscu powodują reakcję — zmianę stanu modelu, wyrażoną zmianą wartości standardowych atrybutów numerycznych i logicznych.

Należy podkreślić, że z punktu widzenia przebiegu symulacji interpretacja nadawana zadaniom jest nieistotna — ważne są jedynie ich rozkłady czasowe oraz priorytety (istnieje 8 klas priorytetów). Zadania mogą być wprowadzane do modelu za pośrednictwem bloków generacyjnych lub z zewnątrz, za pośrednictwem taśmy magnetycznej na podstawie uprzednio zapisanej informacji ustalającej ich rozkład czasowy i podstawowe atrybuty takie, jak priorytet i zbiór parametrów. Każde zadanie występujące w procesie symulacji należy do jednego lub, w szczególnym przypadku, jednocześnie do kilku zbiorów. Zbiory te są uporządkowane według określonych zasad. Łańcuchowa organizacja zadań w zbiorze umożliwia identyfikację każdego elementu. Łańcuchy (zbiory) zadań określają harmonogram zdarzeń w procesie symulacji. Organizacja zbiorów, decyzje dotyczące przenoszenia poszczególnych zadań ze zbioru do zbioru i dołączania nowych zadań podlegają programowi sterowania.

URZĄDZENIE jest zdefiniowane jako wielkość (element wyposażenia) o jednostkowej pojemności, tzn. taki obiekt, który może być w każdej chwili czasowej wykorzystywany tylko przez jedno zadanie. Zatem w modelu urządzenie może reprezentować np. kanał maszyny cyfrowej, jednostkę centralną itp. Zadanie wykorzystujące urządzenie w określonej chwili czasowej może podlegać PRZERWANIOM, tzn. może zostać usunięte z tego urządzenia przez inne zadanie z wyższym priorytetem.

MAGAZYN jest zdefiniowany jako wielkość (obiekt), która może być wykorzystywana jednocześnie przez wiele zadań, przy czym ilość tych zadań nie może przekraczać ustalonego poziomu — pojemności magazynu. Zatem magazyn może reprezentować w modelu np. pamięć maszyny cyfrowej, magazyn towarów itp. Standardowymi atrybutami numerycznymi magazynów są: średnie zawartości, maksymalne zawartości, średnie czasy pobytu zadania itp.

PRZEŁĄCZNIKI LOGICZNE są wykorzystywane w modelu do reprezentowania stanów binarnych, pamiętania określonych wielkości logicznych.

PAMIĘCI są wielkościami pomocniczymi wykorzystywanymi do przechowywania wartości standardowych atrybutów numerycznych, współczynników, wyników pośrednich.

ZBIORY UŻYTKOWNIKA służą do modelowania dowolnych procesów oczekiwania. Organizacja tych zbiorów oraz momenty dołączania lub odłączania zadań są w każdym przypadku ustalane przez programistę. Zadania tworzące zbiory użytkownika są czasowo wyłączone spod kontroli programu sterującego.

TABELE służą do zbierania informacji statystycznych o rozkładach wartości wybranych atrybutów numerycznych oraz badania charakteru tych rozkładów.

Przy budowie modelu często pojawia się konieczność wyrażania złożonych zależności arytmetycznych bądź logicznych między standardowymi atrybutami numerycznymi i logicznymi. W tym celu wprowadzono trzy podstawowe rodzaje zmiennych:

— zmienne arytmetyczne
— zmienne logiczno-numeryczne
— zmienne boolowskie.

Wartościami zmiennych arytmetycznych są wartości wyrażeń arytmetycznych zbudowanych z liczb i atrybutów numerycznych (w tym innych zmiennych).

Przykład

$$V10 = ((Q9 + 3) * 100) + FN4 : 15$$

gdzie : oznacza operację brania reszty modulo, a * operację mnożenia. Liczba poziomów nawiasów jest nieograniczona.

Zmienne logiczno-numeryczne służą do przechowywania wartości otrzymanych przez wybór jednej z dwóch wartości numerycznych na podstawie stanu pewnego atrybutu logicznego.

Przykład

$$VLN5 = VB1, 50, 100$$

W zależności od wartości logicznej zmiennej logicznej *VB*1 wartością zmiennej *VLN*5 będzie 50 albo 100.

$$VLN1 = LS1, VLN2, VLN3$$

Wartością zmiennej *VLN*1 jest w zależności od stanu przełącznika logicznego *LS*1 albo *VLN*2, albo *VLN*3. Wartością **zmiennej boolowskiej** jest wartość logiczna wyrażenia boolowskiego budowanego ze standardowych atrybutów logicznych (nie wykluczając innych zmiennych boolowskich) oraz z relacji między atrybutami numerycznymi.

Przykład

$$VB1 = NEG\ SE5\ AND\ V3\ G\ V2 + 1$$

Odwołanie do zmiennej *VB*1 powoduje wyznaczenie jej wartości logicznej jako koniunkcji zanegowanej wartości logicznej atrybutu logicznego *SE*5 i relacji między zmiennymi arytmetycznymi (wartość *V*3 większa od *V*2+1).

Wiele związków wewnątrz systemu daje się przedstawić w postaci zależności między dwiema zmiennymi. Aby umożliwić zapis związku tego typu, wprowadzono **FUNKCJE.** Każda funkcja wiąże argument, którym jest pewna zmienna niezależna należąca do symulowanego modelu ze zmienną zależną — wartością funkcji. Argumentem funkcji może być dowolny atrybut numeryczny, nie wykluczając innej funkcji. Funkcje znajdują zastosowanie m.in. do generowania zmiennych losowych o dowolnym rozkładzie prawdopodobieństwa. W takich przypadkach argumentem funkcji jest zmienna losowa o rozkładzie równomiernym, wartością zaś funkcji — zmienna losowa o żądanym rozkładzie prawdopodobieństwa. W języku ZAM-GPSS istnieją dwa rodzaje funkcji:

— funkcje numeryczne
— funkcje boolowskie.

Funkcje numeryczne zadawane są w postaci listy par : argument, wartość funkcji. Na liście tej mogą występować zarówno liczby, jak i standardowe atrybuty numeryczne (w szczególności zmienne). Można traktować te listy jako funkcje typu schodkowego (dyskretne) lub jako funkcje przedziałami liniowe (automatyczna interpolacja liniowa). Istnieje wygodny sposób zapisu funkcji zadanych na kolejnych liczbach naturalnych.

Przykłady funkcji

$FN1 = X3, CN2, 1, 4, 3, 7$	'Funkcja ciągła numeryczna (typ funkcji jest określony przez specyfikator *CN*), złożona z dwóch par liczb. Argumentem funkcji jest atrybut X3'
$FN2 = S3, LA3, 5, 1, 15$	'Funkcja zadana w postaci listy wartości funkcji (typ *LA*) dla argumentów naturalnych. Argument funkcji jest określony za pośrednictwem atrybutu *S*3'

Funkcje boolowskie reprezentują związki między argumentami numerycznymi a wartościami funkcji, które zawsze są wartościami typu logicznego.

Przykład

$$FNB1 = X1, D3, 1, LS1, 4, SE2, 7, FNU3$$

Funkcja boolowska *FNB*1 jest funkcją zadaną za pośrednictwem par : argument numeryczny, atrybut logiczny. Argumentem funkcji jest atrybut X1.

Przykład programu

Aby zilustrować sposób wykorzystania w modelu funkcji, parametrów i atrybutów numerycznych, rozpatrzmy model sklepu samoobsługowego. Klienci przybywający z określoną częstotliwością są zobowiązani pobrać koszyk przed wejściem do sklepu. Ilość koszyków jest oczywiście ograniczona. Przyjmiemy, że w przypadku, gdy nie ma wolnych koszyków, klienci nie czekając opuszczają sklep. Po pobraniu koszyka klienci dokonują zakupów, po czym ustawiają się w kolejce do jednej z pięciu kas. Klient opuszczający sklep po dokonaniu zakupów zwraca koszyk.

```
SYMULACJA
. . . . . .
. . . . . .
'DEFINICJE WIELKOŚCI'
'DEFINICJE POJEMNOŚCI MAGAZYNÓW'
S1 = 5                  'ILOŚĆ KAS'
S2 = 50                 'ILOŚĆ KOSZYKÓW'
. . . . . .
. . . . . .
'DEFINICJE FUNKCJI'
FN1 = . . . . . 'FUNKCJA OKREŚLAJĄCA
                ROZKŁAD CZASOWY
                PRZYBYWANIA KLIENTÓW
                DO SKLEPU'
FN2 = . . . . . 'FUNKCJA OKREŚLAJĄCA
                ILOŚĆ TOWARÓW (LISTA
                ZAKUPÓW) ZAKUPYWA-
                NYCH W SKLEPIE'
FN3 = . . . . . 'FUNKCJA OKREŚLAJĄCA
                CZAS ZAKUPÓW'
. . . . . .
. . . . . .
```

```
MODEL
GENERUJ, 36, FN1        'ZADANIA REPREZENTUJĄ KLIENTÓW. ARGUMENTY BLOKU OKREŚLAJĄ CZĘSTOTLIWOŚĆ PRZYBYWANIA KLIENTÓW — PRZYJĘTO, ŻE ŚREDNIA LICZBA KLIENTÓW PRZYBYWAJĄCYCH DO SKLEPU W CIĄGU GODZINY JEST RÓWNA 36'
SKOCZ B, ALFA           'SPRAWDZENIE CZY SĄ WOLNE KOSZYKI'
WPROWADŹ, 1             'POBRANIE KOSZYKA — MAGAZYN 1 REPREZENTUJE W MODELU ZAPAS KOSZYKÓW'
PODSTAW, 1, FN2         'USTALANIE LISTY ZAKUPÓW — FUNKCJA FN2 OKREŚLA ROZKŁAD PRAWDOPODOBIEŃSTWA ILOŚCI ZAKUPYWANYCH TOWARÓW'
OPÓŹNIJ, FN3            'CZAS TRWANIA ZAKUPÓW — FUNKCJA FN3 OKREŚLA ROZKŁAD CZASÓW TRWANIA ZAKUPÓW'
KOLEJKA, 1              'OCZEKIWANIE W KOLEJCE DO KASY'
WPROWADŹ, 2             'MAGAZYN 2 REPREZENTUJE W MODELU GRUPĘ KAS DOSTĘPNYCH DLA KLIENTÓW'
WYJDŹ, 1                'OPUSZCZENIE KOLEJKI'
OPÓŹNIJ, 3, 1           'CZAS SPĘDZONY PRZEZ KLIENTA PRZY KASIE'
POBIERZ, 2              'ZWOLNIENIE KASY'
TABELARYZACJA, 1        'TABELARYZACJA CZASU POBYTU KLIENTA W SKLEPIE'
POBIERZ, 1              'ZWROT KOSZYKA'
ALFA : NISZCZ, 1        'NISZCZENIE ZADANIA — KLIENT OPUSZCZA SKLEP'
KMODEL
START, 1000             'DYREKTYWA STARTU SYMULACJI'
```

Wynikiem procesu symulacji jest pełny raport o stanie końcowym modelu obejmujący wartości wszystkich standardowych atrybutów wielkości zdefiniowanych w problemie oraz tabela rozkładów czasów pobytu klientów w sklepie (zadań w modelu).

Powtarzając symulację dla różnych wartości parametrów można określić warunki dla najszybszej obsługi klientów (np. można określić wymaganą liczbę koszyków i kas).

4. Informacje eksploatacyjne

Język symulacyjny ZAM-GPSS został zrealizowany w Instytucie Maszyn Matematycznych, na komputerze ZAM-41.

Wymagany zestaw maszyny:

a) pamięć operacyjna — 20 *K* słów

b) pamięć bębnowa — minimum 30 000 słów

c) drukarka wierszowa

d) dowolny zestaw urządzeń wejściowych (taśmy magnetyczne, czytniki taśmy perforowanej, czytniki kart, monitor dalekopisowy). Istnieje możliwość zmiany urządzenia wejściowego w czasie wprowadzania programu źródłowego lub informacji sterujących przebieg symulacji (dyrektywa zmiany WEJŚCIA)

e) program symulacyjny może współpracować z czterema jednostkami taśm magnetycznych. Taśmy magnetyczne mogą być wykorzystywane do tworzenia biblioteki programów lub jako zewnętrzne źródło informacji dla programu symulacyjnego.

Dane eksploatacyjne:

a) maksymalna długość programu symulacyjnego jest ograniczona jedynie pojemnością pamięci operacyjnej. Dla przyjętego zestawu komputera liczba bloków — operatorów może przekraczać 1000, a liczba zadań aktywnych — 700

b) średnia szybkość działania — 100 do 300 bloków na sekundę

c) ZAM-GPSS działa pod kontrolą systemu operacyjnego SO 141.

5. Zakończenie

Kilkumiesięczna eksploatacja próbna języka ZAM-GPSS wykazała jego przydatność jako praktycznego narzędzia ułatwiającego prace badawcze i projektowe. Istniejące ograniczenie pojemności pamięci operacyjnej maszyny cyfrowej ZAM-41 umożliwia uruchomienie programów symulacyjnych średniej wielkości. Pierwszym zastosowaniem praktycznym języka ZAM-GPSS była symulacja koncepcji realizacyjnej i działanie programów wykonawczych tego języka. Wstępna wersja programu posłużyła jako narzędzie do sprawdzenia i modyfikacji przyjętych założeń realizacyjnych, optymalizacji czasowej rozwiązań wewnętrznych.

BIBIOGRAFIA

[1] Język ZAM—GPSS. Instrukcja programowania. IMM.

[2] GPSS/360. User's manual.

[3] G. Gordon: System Simulation.

Uwaga, Czytelnicy i Prenumeratorzy!

podajemy nowy adres

Redakcja czasopisma „INFORMATYKA"

Warszawa, ul. Jasna 14/16, pokój 332 na III piętrze

tel. 26-82-61, w. 285

THOMAS E. CHEATHAM, Jr

681.322.06

Dotychczasowy rozwój języków programowania

Część I

Artykuł ten jest opracowany na podstawie referatu wygłoszonego przez T. E. Cheathama (jr) na Kongresie IFIP w 1971 r. Autor próbuje ocenić sytuację w dziedzinie języków i systemów programowania i na tej podstawie przewidzieć, czego można się spodziewać w tej dziedzinie w ciągu najbliższego dziesięciolecia.

1. Wprowadzenie

W ubiegłym dziesięcioleciu byliśmy świadkami olbrzymiego postępu dokonanego w rozwoju, zastosowaniach i zrozumieniu języków oraz systemów programowania, a także rozmaitych środków ułatwiających definiowanie, analizę, syntezę i interpretację tych języków. Artykuł ten jest próbą oceny sytuacji w dziedzinie języków programowania, systemów, w których są one realizowane i związanych z nimi środków programowych. Chcemy umieć odpowiedzieć sobie na pytania: gdzie jesteśmy, w jaki sposób doszliśmy do obecnych wyników i czego możemy się spodziewać po następnym dziesięcioleciu, a także sprecyzować niektóre z najważniejszych pojawiających się problemów.

Omawiając niektóre z dawniej nakreślonych celów rozwoju języków programowania zastanowimy się, w jakim stopniu je osiągnięto. Szczególną uwagę zwrócimy na zagadnienie: czego nauczyliśmy się w ubiegłym okresie i co możemy, a nawet, moim zdaniem, powinniśmy robić w najbliższym dziesięcioleciu. Oczywiście, trudno dokładnie to przewidzieć. Przyjęcie i szerokie stosowanie języków programowania zależy w ogromnej mierze od nastawienia wytwórców maszyn i oprogramowania, zespołów użytkowników, komitetów ustalających standardy, a nawet rządów. Warunkują je również: istnienie niezawodnych, dobrze udokumentowanych i poręcznych realizacji danych języków dla rozmaitych maszyn cyfrowych, określenie zadowalających standardów, dostępność odpowiednich podręczników, utworzenie się silnych grup użytkowników z rozmaitych dziedzin zastosowań, istnienie dobrej biblioteki podprogramów standardowych, dostępnej dla wszystkich użytkowników danego języka. Nie można z góry założyć, w jakim stopniu będą się rozwijać wymienione czynniki dla omawianych poniżej języków i nie zamierzamy tego czynić.

Artykuł składa się z czterech rozdziałów oraz z bibliografii. W rozdziale 2 zajmiemy się rozwojem języków programowania i bazowych dla nich systemów programowania dokonanym w latach 60; w rozdziale 3. omówimy rozwój narzędzi służących do definiowania, analizy, syntezy i interpretacji języków programowania. W rozdziale 4. przedstawimy nasze przewidywania — co możemy i powinniśmy robić w następnym dziesięcioleciu oraz omówimy pewne ważne zagadnienia, z których trzeba sobie zdać sprawę, jeżeli chcemy, aby dalszy rozwój języków programowania w latach 70 przebiegał gładko.

2. Dotychczasowy rozwój języków i systemów programowania

Nasze rozważania podzielimy na pięć punktów. W pierwszym omówimy kilka popularnych niekonwersacyjnych języków programowania, w następnym zaś kilka języków właściwych dla zastosowań konwersacyjnych. Przyporządkowanie danego języka do jednej z tych dwóch grup nie zależy od tego, jak bywa on zazwyczaj stosowany, ponieważ jest to fakt raczej bez znaczenia i w przypadku konkretnej realizacji języka zaciemniający istotną różnicę. Dokonując podziału zwracamy uwagę na takie cechy, jak wrażliwość języka na rozgraniczenie etapów: „czasu tłumaczenia", „czasu wprowadzania" i „czasu wykonywania" dla danego programu lub, podchodząc do tego zagadnienia z innej strony, wrażliwość na ograniczenia nakładane na zmienianie powiązań nazw z deklaracjami typów wielkości i ich wartościami. W p. 2.3. zajmiemy się rozwojem języków specjalnych, w p. 2.4. zaś omówimy aktualny stan w dziedzinie języków „rozszerzalnych". Ostatni punkt tej części będzie dotyczył obecnej sytuacji w dziedzinie sieci maszyn.

2.1. Popularne języki niekonwersacyjne

2.1.1. FORTRAN

FORTRAN jest z pewnością obecnie najważniejszym językiem programowania dla zagadnień naukowych i prawdopodobnie pozostanie nim jeszcze przez jakiś czas. Wprawdzie od czasu do czasu słyszy się przepowiednie o jego rychłej śmierci, ale (jak zauważył A. J. Perlis) „FORTRAN nie jest kwiatem, lecz chwastem — jest odporny, kwitnie sporadycznie i porasta każdą maszynę". Jest to pierwszy język programowania, którego nauczono tak wielu ludzi. Prawie każda maszyna ma przynajmniej jeden, a zazwyczaj kilka zrealizowanych wersji FORTRAN-u. W niektórych jego realizacjach np. PUFT, WATFOR i WATFIWE [1, 2] koszt przetłumaczenia i wykonania małych zadań nie przekracza kilku centów.
Najważniejszy wniosek, jaki można wyciągnąć z historii FORTRAN-u, dotyczy nie tyle jego trwałości wynikłej z powszechnego przyjęcia go i stosowania, ile przede wszystkim jego standaryzacji. Od kilku lat istnieje „standard" FORTRAN-u, lecz wciąż powstają godne uwagi odmiany tego języka w rozmaitych realizacjach. Zbadano [3], że np. w sześciu różnych realizacjach FORTRAN-u dla średnich i dużych maszyn jedyną instrukcją całkowicie im wspólną była CONTINUE. Pomimo to FORTRAN okazał się ogromnie przydatny do przenoszenia programów z jednej maszyny na inną, oczywiście, jeżeli się ostrożnie korzysta z pewnych unikalnych własności niektórych realizacji. Ponadto FORTRAN można realizować zarówno dla niedrogich i o małej pamięci „minikomputerów", jak i największych istniejących zestawów maszyn cyfrowych.

2.1.2. COBOL

COBOL jest zapewne drugim po FORTRAN-ie najszerzej stosowanym i najbardziej pożytecznym językiem programowania. W przeciwieństwie do FORTRAN-u istnieje rzeczywiście standardowy podzbiór COBOL-u i obecnie wiele maszyn ma standardowe wersje tego języka pozwalające na wymianę programów bez wprowadzania do nich zmian (a czasem nawet wymianę poszczególnych plików).

Dla większości użytkowników naukowych i badaczy języków programowania COBOL nie jest zbyt interesujący. Niemniej wciąż jeszcze jest on jednym z niewielu języków pozwalających opisywać (w przeciwieństwie do deklarowania lub manipulowania) za-

równo dane dla programu, jak i wyposażenie maszyny właściwe dla wykonywania danego programu. COBOL jest również jednym z niewielu języków umożliwiających segmentację programu oraz zawierających tak złożone operatory jak „posortuj plik" lub „utwórz sprawozdanie".

2.1.3. ALGOL-60

Wprawdzie nie jest to język najczęściej używany (zwłaszcza w Stanach Zjednoczonych), niemniej powinien być uważany za najważniejszy ze wszystkich języków programowania, jakie dotychczas powstały. Pojawienie się ALGOL-u z pełnym opisem syntaktycznym, rozsądnym (choć nieformalnym) opisem semantycznym i licznymi niejednoznacznościami spowodowało ożywione poszukiwania modeli dla opisów składni i semantyki i inne badania uwieńczone ciekawymi wynikami. Klarowność i symetria struktury ALGOL-u-60, a także dokonany przez jego autorów trafny wybór poszczególnych reprezentacji dla programów (wzorcowa, publikacyjna i liczne wersje realizacyjne) dały w wyniku pewien model, wywierający ogromny wpływ na wszystkie następnie tworzone języki programowania. Na przykład jako wzór dla języka PL/1, projektowanego głównie w IBM, wybrano jednak ALGOL-60, a nie FORTRAN.

2.1.4. PL/1

PL/1 był pierwszą (a raczej pierwszą, która przetrwała) próbą utworzenia języka uniwersalnego, nadającego się w sposób naturalny lub przynajmniej rozsądny do różnorodnych zastosowań programowych. Spodziewano się, że będzie on odpowiedni dla zasadniczo wszystkich potrzeb obliczeniowych. Dla wielu przyczyn języka PL/1 nie przyjęto powszechnie. Najważniejsze z nich to: 1) żadna ważna grupa użytkowników nie nalegała, aby uczynić go językiem standardowym, dostępnym w rozmaitych zestawach maszyn cyfrowych; 2) PL/1 jest językiem ogromnym i prawdopodobnie jego realizacja byłaby niemożliwa dla maszyn średnich i małych; 3) pozostali, oprócz IBM, wytwórcy maszyn odnosili się do tego języka niechętnie, nie propagując go i nie realizując na ogół; 4) do niedawna nie było żadnego podzbioru PL/1 do przetwarzania danych dla celów naukowych lub jakichś innych zastosowań odpowiednich dla maszyn szybkich, lecz ze skromnym wyposażeniem dodatkowym.

PL/1 był jednym z pierwszych języków uniwersalnych (prawdopodobnie pierwszy był LISP), dla którego stworzono pełny formalny model semantyczny [4, 5, 6, 7], jednakże wydaje się, że będzie on przydatny jedynie dla szczupłej garstki specjalistów, ponieważ nie korzysta z niego nawet grupa ludzi zajętych tworzeniem standardowej wersji PL/1 [8].

2.1.5. ALGOL-68

Z zamierzenia ALGOL-68 miał być podobnie jak PL/1 językiem uniwersalnym, lecz jego struktura okazała się dużo bardziej klarowna i harmonijna (podobną sytuację mamy w przypadku ALGOL-u-60 i FORTRAN-u). Powstanie ALGOL-u-68 przyniosło uogólnienie pojęć danych i operacji oraz uproszczenie budowy języka ALGOL-60 i przyczyniło się do doskonalszego i subtelniejszego rozumienia języków programowania w ogólności. Z tego właśnie względu ALGOL-68 jest szczególnie ważnym wydarzeniem.

W chwili obecnej powstaje kilka realizacji ALGOL-u-68 oraz istnieje kilka translatorów już pracujących. Podobnie jak PL/1, ALGOL-68 jest językiem obszernym i jego translatory oraz zespoły procedur realizujących wykonywanie programów wynikowych muszą być wielkimi programami, należy zatem się spodziewać, że z tego języka można będzie korzystać jedynie w dużych i średnich zestawach maszyn cyfrowych.

ALGOL-68, utworzony wprawdzie przez komitet, nie uzyskał poparcia żadnego wytwórcy maszyn ani żadnej silnej i poważnej grupy użytkowników. Nie jest też przeznaczony do zastosowań konwersacyjnych. Nie wydaje się zatem prawdopodobne, aby mógł on spotkać się z powszechnym przyjęciem, jednakże z pewnością będzie przez wiele lat służył jako ważny model języka programowania, podobnie jak ALGOL-60 i jego wpływ będzie raczej teoretyczny niż praktyczny.

2.2. Uniwersalne języki konwersacyjne

2.2.1. BASIC

Język BASIC jest zasadniczo wzorowany na FORTRAN-ie, a przy tym tak zmodyfikowany (pod względem budowy programu, uruchamiania i wykonywania programów), aby nadawał się do korzystania w systemach konwersacyjnych. Podobnie jak FORTRAN, spotkał się z powszechnym przyjęciem i przypuszczalnie przez jakiś czas jeszcze pozostanie najbardziej popularnym językiem używanym w systemach konwersacyjnych.

2.2.2. APL

Najbardziej interesujące jest powstanie i przyjęcie się języka APL, przeznaczonego specjalnie do zastosowań konwersacyjnych, posiadającego wiele cech szczególnie uwypuklających to jego przeznaczenie. APL różni się od większości innych języków konwersacyjnych tym, że urządzeniem końcowym w przypadku jego czystej wersji jest maszyna cyfrowa IBM 2741 oraz przede wszystkim specjalnym zbiorem symboli, zawierającym litery greckie i pewną liczbę niestandardowych symboli operatorów.

Zasadniczo APL przeznaczono do wykonywania działań na wektorach i macierzach, toteż zawiera on bogaty zbiór operatorów powodujących tworzenie i przekształcanie macierzy. Język ten wprowadza inny styl programowania niż utworzony pod wpływem ALGOL-u czy FORTRAN-u itp. i to zarówno w przypadku stosowania macierzy jako reprezentacji dowolnych struktur danych, jak i w stosunku do struktury sterowania wykonanie algorytmu. Dzięki temu można częstokroć zakodować duże algorytmy w pojedynczych instrukcjach APL (np. obliczyć i wydrukować wszystkie permutacje skończonego zbioru liczb, obliczyć i wydrukować wszystkie czynniki pierwsze jakiejś liczby itp.).

Być może najważniejszymi wnioskami wypływającymi z doświadczeń z językiem APL są: uświadomienie sobie roli, jaką odgrywa cały system programowania w stosunku do realizowanego w nim języka konwersacyjnego oraz zwrócenie uwagi na właściwy dobór zbioru symboli dla danego języka (APL przy zastosowaniu dalekopisu jako urządzenia końcowego jest daleko mniej przydatny niż przy korzystaniu z maszyny IBM 2741). Warto też podkreślić fakt, że możliwość korzystania z dużej rozmaitości operatorów łączonych za pomocą bardzo prostej składni (w APL wszystkie operatory mają tę samą hierarchię) daje w wyniku bardzo silne środki programowe.

2.2.3. PPL

Język ten jest rozszerzeniem i uogólnieniem APL [9, 10]. Wprowadzono w nim bogaty zbiór typów danych podstawowych oraz stworzono środki umożliwiające definiowanie nieskończonej rozmaitości struktur danych. Przewidziano ułatwienia pozwalające definiować szeroki zakres rodzajów sterowania (np. równoległe ścieżki sterowania korzystając z semaforów Dijkstry w celu ich koordynacji, procedury skojarzone itp.), a także możliwość definiowania nowych operatorów dodatkowych. PPL powstał właściwie z APL drogą rozszerzania go i wcielania rozmaitych pomysłów pojawiających się w ostatnich latach w dziedzinie tzw. „języków rozszerzalnych" przy jednoczesnym troskliwym chronieniu struktury i przejrzystości właściwej APL. Wprawdzie nie wiadomo

jeszcze, czy PPL stanie się językiem popularnym, niemniej jest on pierwszym produktem powstałym z poszukiwań języków rozszerzalnych, mających wielką przyszłość przed sobą.

2.2.4. LISP

Język LISP, pierwotnie zaprojektowany przez J. Mc Carthy, jest na równi z ALGOL-em-60 najważniejszym językiem z powstałych dotychczas języków programowania. Powstały przed dziesięciu laty był prawdopodobnie pierwszym językiem programowania mającym formalny model semantyczny, przy tym skonstruowany w samym LISP-ie. Mimo że od lat używano go w systemach niekonwersacyjnych, umieściliśmy go w tym punkcie, ponieważ prawdopodobnie bardziej niż inne omawiane tu języki ma strukturę odpowiednią do zastosowań konwersacyjnych. Zawdzięcza to brakowi ograniczeń dotyczących zmiany powiązań nazw z typami wielkości i ich wartościami.

LISP okazał się również szczególnie odpowiedni (jeżeli pominiemy sprawę oszczędności miejsca i czasu) jako baza dla tworzenia dużych systemów, jak np. MATHLAB i PILOT, systemów informacyjnych w językach naturalnych oraz rozmaitych systemów sterowania „robotami".

Próbowano kilkakrotnie „dolepić" do typów danych, operacji i zasadniczej struktury języka LISP rodzaje danych, operacje i własności spotykane w językach podobnych do ALGOL-u, żadna z tych prób jednakże nie powiodła się. Kilka lat temu zarzucono najambitniejszą z tych prób — projekt LISP 2. Nie oznacza to, że nie było przypadków zlepienia pewnych własności typowych dla LISP-u z możliwościami języków popularnych, np. takimi językami są FORMULA-ALGOL, TREET, SLIP, itp. Nie powstał jednak dotychczas żaden język ani system programowania łączący w pełni cechy LISP-u i języków ALGOLO-podobnych, z wyjątkiem systemu ECL [11], o którym będzie jeszcze mowa w p. 2.5.

2.3. Języki specjalne

W ubiegłym dziesięcioleciu można było zaobserwować pojawienie się tzw. języków specjalnych czy też języków do zastosowań specjalnych *(special purpose languages)*. Niektóre z nich powstały jako uzupełnienia (ale nie „rozszerzenia", gdyż ten termin należy zastrzec dla innych celów) istniejących już popularnych języków (np. SLIP, FORMULA-ALGOL, METEOR), inne powstały samodzielnie, np. SNOBOL, APT, COGO, SAIL, PLANNER itp.

Być może niektóre z tych języków powstały z prywatnych pobudek ich twórców, lecz były też i głębsze przyczyny pojawienia się większości z nich. Języki popularne lub, mówiąc precyzyjniej, ich poszczególne realizacje nie są ani wygodne, ani oszczędne dla pewnych problemów. Przejawów tego jest wiele — czasem chodzi o ułatwienia przy pisaniu (i czytaniu) programów, czasem najważniejsza jest szybkość tłumaczenia i wykonywania programów lub też wielkość translatora, kiedy indziej zależy na umożliwieniu pracy konwersacyjnej. Wydaje się szczególnie ważną sprawą zasygnalizowanie popytu na takie języki, w których programy dla pewnych zagadnień można pisać w sposób „naturalny" (lub może raczej chodzi o to, aby mogli je pisać „programiści-amatorzy"). Wprawdzie większość języków programowania jest teoretycznie odpowiednia dla większości problemów przeznaczonych do zaprogramowania, jednakże ten argument nie czyni ich przydatnym narzędziem w praktyce. W istocie wielu ludzi dzisiaj uważa, że w następnym dziesięcioleciu będzie w dalszym ciągu przybywać języków specjalnych; w rozdziale 4. tego artykułu omówimy pewne środki ekonomiczne podtrzymujące ich rozwój.

2.4. Rozszerzalne języki programowania

Językami rozszerzalnymi *(extensible programming languages)* nazywamy takie, które składają się z języka bazowego (czy też rdzenia języka) oraz korzystają z systemu programowania umożliwiającego rozszerzenie lub modyfikowanie następujących cech języka: 1) składni programów, 2) operatorów języka, 3) struktur danych oraz ich wewnętrznej reprezentacji i 4) rodzajów sterowania.

Językami rozszerzalnymi zaczęto zajmować się w połowie lat 60 w związku z uzyskanymi doświadczeniami przy konstruowaniu języków uniwersalnych, takich jak PL/1, FORMULA-ALGOL czy LISP 2. Prace przebiegały w małych grupach badaczy i dotychczas nie odegrały dużego znaczenia praktycznego, lecz przewiduje się ich znaczny wpływ na rozwój języków w następnym dziesięcioleciu (patrz rozdz. 4 artykułu).

Można wyróżnić trzy etapy w dotychczasowych pracach nad językami rozszerzalnymi: 1) tworzenie syntaktycznych „dolepek" do popularnych języków, 2) wprowadzanie nieznacznych zmian do (zazwyczaj) ALGOL-u-60 po to, aby jego zasadnicza struktura i pojęcia pierwotne dopuszczały rozszerzalność i 3) projektowanie całkowicie nowych języków przy zwróceniu dużej uwagi na ich strukturę, ogólność i efektywność wymagane przy wprowadzaniu rozszerzeń. Pierwsze dwa etapy trwały dość krótko. Po wprowadzeniu do rozmaitych języków kilku rozszerzeń syntaktycznych zwanych zazwyczaj makroinstrukcjami składniowymi lub obliczanymi zauważono wkrótce, że jest to bardzo słaby aparat, ponieważ języki bazowe nie miały wielu cech istotnych dla efektywnej rozszerzalności. Powstało też następnie kilka przebudowanych wersji ALGOL-u, np. ALGOL-D [15], GPL [16, 17], IMP [18], ale większość z nich zrealizowano tylko dla niewielu maszyn.

Zasadniczy postęp w tej dziedzinie datuje się od czasu, gdy pod wpływem solidnych badań zarówno struktury języków programowania, jak i ich formalnych modeli semantycznych zaczęto realizować całkowicie nowe języki. Powstały wówczas: BASEL [19], PPL [9], GEDANKEN [20], ECL [11], PASCAL [21] itp. W tej chwili jest jeszcze za wcześnie na ocenę, czy te języki odpowiadają w pełni wymaganiom i czy się utrzymają, chociaż wydaje się, że PPL przyjął się dość szybko.

Na tle szybkiego rozwoju omawianych języków można było zaobserwować pewną krystalizację pojęcia określanego terminem „język rozszerzalny"[1]), które można scharakteryzować w następujących siedmiu punktach:

(1) Język taki ma formalny model semantyczny i prawdopodobnie powstał przez konstruowanie najpierw tego modelu, dla którego cechy syntaktyczne rozważano jako wtórne.

(2) Model semantyczny sugerował efektywną realizację dla nowoczesnej maszyny cyfrowej.

(3) Język taki ma pewną liczbę pierwotnych typów danych (np. całkowite, rzeczywiste, boolowskie, znakowe) oraz stosowane rekursywnie środki pozwalające wprowadzać nowe typy danych przy pomocy jednorodnych ciągów wielkości (zwanych czasem strukturami lub zbiorami), wskaźników, procedur, unii uprzednio określonych typów danych itp. Z każdym typem danych jest związany aparat pozwalający konstruować przykłady, wydzielać części danych i sprawdzać, czy dany obiekt jest określonego typu.

(4) Język taki ma kilka operatorów pierwotnych oraz środki dla definiowania nowych operatorów i modyfikacji lub rozszerzania starych definicji.

(5) Język taki ma pewną liczbę prostych mechanizmów sterowania (np. wykonywanie sekwencyjne, iteracja, odwołania do procedury, stosowanie przerwań itp.) oraz środki dla konstruowania nowych rodzajów sterowania, w tym procedur skojarzonych *(coroutines)*, ścieżek równoległych, semaforów Dijkstry, wy-

[1]) Obecnie jedynym znanym autorowi językiem spełniającym wszystkie przytoczone kryteria jest ECL.

rażeń obliczanych w sposób ciągły, częściowego porządkowania itp.

(6) Składnia takiego języka powinna być przedmiotem modyfikacji użytkownika.

(7) Taki język powinien umożliwiać użytkownikowi „zamrażać" powiązania nazw z typami wielkości i ich wartościami, natomiast translator powinien z programu źródłowego i specyfikacji „zamrożonych powiązań" produkować pewną reprezentację [2]) tego programu przy pomocy jego struktur danych i ciągów rozkazów. A więc zaciera się tutaj granica między pojęciami „tłumaczenie-wprowadzanie" a „wykonywanie".

2.5. Sieci maszyn cyfrowych

W ubiegłych latach powstało kilka tzw. sieci maszyn *(networks of computers)*. Większość z nich to względnie skromne organizacje utworzone w celu znacznego przyspieszenia przekazywania danych między dwiema lub trzema maszynami znajdującymi się w niewielkiej odległości, lub między dużą maszyną i kilkoma odległymi stacjami przyjmującymi dane wejściowe, często po prostu małymi maszynami traktowanymi jako urządzenie wejściowe dla dużej maszyny. Spośród rozmaitych sieci maszyn obecnie jedna — ARPA Network — wyróżnia się znacznym skomplikowaniem budowy. Ma ona już ponad dwieście węzłów rozrzuconych na obszarze Stanów Zjednoczonych — od Kalifornii do Massachusetts, a wkrótce dołączą do nich inne, w Anglii i Norwegii. Każdy węzeł składa się z łącza IMP (Interface Message Processor) powiązanego z jedną lub kilkoma bazowymi maszynami cyfrowymi o bardzo różnych rozmiarach, od PDP-11 do ILLIAC-IV. Łącza IMP są powiązane między sobą specjalnymi liniami telefonicznymi. Porcje przepływu danych (strumienie bitów) są rzędu 50 000 bitów/sek, a czas przekazywania informacji od wysyłającego do odbiorcy mierzy się w dziesiątkach milisek. Koszt przesyłania danych jest znacznie mniejszy niż jeden dolar za megabit. Mimo że ARPA Network znajduje się jeszcze w początkowym stadium rozwoju, jej istnienie wykazało możliwość realizowania podobnych zamierzeń. Wydaje się rzeczą pewną, że sieci maszyn odegrają ogromną rolę w następnym dziesięcioleciu i to nie tylko ze względów finansowych.

ARPA Network [22] jest prototypem nowego rodzaju maszyny cyfrowej, który może mieć zasadniczy wpływ na języki i systemy programowania, a nawet na rozwój informatyki w najbliższym dziesięcioleciu. Oczywiście, jednym ze sposobów korzystania z niej jest po prostu przesyłanie programów z danymi lub oddzielnie do rozmaitych odległych miejsc i tłumaczenie oraz wykonywanie ich za pomocą translatorów w poszczególnych maszynach bazowych. Można jednakże dla każdej takiej sieci maszyn wyobrazić sobie całkiem nowe zastosowania, np. tworzenie wokół niej grup ludzi rozwiązujących wspólnie i równocześnie jakieś zadania. Urzeczywistnienie tego wymaga jednak języka i systemu programowania o własnościach umożliwiających wspólne układanie, sprawdzanie i wykonywanie programów oraz pozwalających tworzyć organizacje ludzkie odpowiednie do stawianych przed nimi zadań. Można sobie wyobrazić istnienie jednego „władcy" i kilku „niewolników", można też budować organizacje, w których wszyscy partycypują równorzędnie przy zachowaniu odpowiednich zabezpieczeń przed mimowolnymi szkodami podczas dokonywania modyfikacji programów lub danych.

Dla takich sieci maszyn szczególnie przydatny jest język i system programowania ECL, powstały przed rokiem w Harvard [11]. Język systemu ECL jest rozszerzalny w sensie opisanym w poprzednim punkcie, a system nie nakłada żadnych ograniczeń na strukturę organizacyjną zrzeszanych w grupy użytkowników, pozwalając im wybierać rodzaj sterowania i sposób utrzymywania łączności odpowiedni do ich potrzeb.

cdn.

Oprac. Jowita Koncewicz

[2]) Mamy więc tutaj na jednym krańcu w pełni interpretacyjną wersję programu, a na drugim „odpowiedź" na sformułowanie za pomocą programu źródłowego zagadnienie; interesuje nas obszar leżący między tymi krańcami.

BIBLIOGRAFIA

[1] P. W. Shantz et al.: WATFOR — The University of Waterloo FORTRAN IV Compiler, C. ACM, 10, 1 (1967).
[2] S. Rosen et al.: PUFT — The Purdue University Fast FORTRAN Translator, CACM 8, 11 (1965).
[3] J. Guy: Program transferability, Proc. Invitational Workshop on Networks of Computers (Oct. 1968).
[4] P. Lucas et al.: Informal introduction to the abstract syntax and interpretation of PL/1, TR 25.083 (IBM Laboratory, Vienna, June 1968).
[5] P. Lucas and K. Wolk: On the formal description of PL/1, (IBM Lab. Vienna, Dec. 1968).
[6] K. Walk et al.: Abstract syntax and interpretation of PL/1, TR 25.082 (IBM Lab.; Vienna, June 1968).
[7] P. Wegner: Theories of semantics, Tech. rep. no. 69/10 (Center for Information Sciences, Brown University, Sept. 1969).
[8] M. D. MacLaren: PL/1 language development in ANSI/X3J1 and ECMA/TC10, SIGPLAN Notices, 5, 12 (Dec. 1970).
[9] E. A. Taft: PPL user's manual (Aiken Comp. Lab., Harvard Univ., Jan. 1971).
[10] T. A. Standish: A preliminary sketch of a polymorphic programming language (Centro de Calculo Electronic, Universidad Nacional de Mexico, June 1968).
[11] B. Wegbreit: The ECL programming system, (Aiken Comp. Lab., Harvard Univ. 1971).
[12] A. J. Perlis: The synthesis of algorithmic systems, Proc. of 21 st National Conf., ACM (Thompson Book Company, Washington, D. C., 1966).
[13] T. E. Cheatham: The introduction of definitional facilities into higher level programming languages, AFIPS Fall Joint Comp. Conf., 29 (1966).
[14] B. M. Leavenworth: Syntax macros and extended translation, Comm. ACM, 9, 11 (Nov. 1966).
[15] B. A. Galler and A. J. Perlis: A proposal for definitions in Algol, Comm. ACM, 10, 4 (April 1967).
[16] P. W. Abrahams: The LISP 2 language and systems, AFIPS proc., Comp. Conf., 29 (1966).
[17] J. V. Garwick: GPL, a truly general purpose language, Comm. ACM, 11, 9 (Sept. 1968).
[18] E. T. Irons: IMP, working paper, no. 229, IDA, Communic. Research Division, Princeton, New Jersey (April 1968).
[19] P. Jorrand and M. M. Hammer: The formal definition of BASEL — Parts 1, 2 and 3 (Massachusetts Comp. Assoc. Inc., Wakefield, Massachusetts, August 1969).
[20] J. C. Reynolds: GEDANKEN, a simple typeless language based on the principles of completeness and the reference concept, CACM, 13, 5 (1970).
[21] N. Wirth: The programming language PASCAL (Edig. Technische Hochschule, Zurich, Nov. 1970).
[22] F. Heart et al.: The interface message processor for the ARPA computer network, AFIPS Proc., Spring Joint Comp. Conf., 36 (1970).
[23] COBOL—1961, Revised specyfications for a common business oriented language (May 1961).
[24] K. E. Iverson: A programming language, (John Wiley and Sons, Inc. New York, 1962).
[25] C. H. Lindsey and S. G. van der Meulen: Informal introduction to Algol 68, (Math. Centrum, Amsterdam, August 1969).
[26] J. McCarthy et al.: LISP 1.5 programmer's manual, (MIT Press, Cambridge, Mass., 1962).
[27] P. Naur ed.: Revised report on the algorithmic language ALGOL 60, Comm. ACM, 6, 1 (January 1963).

OD REDAKCJI

Otrzymaliśmy wiele listów dotyczących terminologii informatyki. W niniejszym zeszycie publikujemy obszerną wypowiedź Stefana Zawadzkiego. Nie zamykamy jeszcze dyskusji na ten temat i prosimy o dalsze uwagi.

Szanowny Panie Redaktorze!

Chciałbym wyrazić najwyższe uznanie dla wysiłku włożonego w opracowanie słowniczka opublikowanego w numerach 5, 6 i 7/71 jako „Projekt normy pojęć podstawowych". Potrzeba uporządkowania terminologii nasuwa się od dawna, gdyż u nas w Polsce — jak dotąd — co maszyna, to inna terminologia. Poszczególne kolektywy związane z różnymi maszynami, tłumacząc przeważnie z jęz. angielskiego instrukcje obsługi komputerów i podręczniki programowania, na własną rękę ustalały i nadal ustalają, wymyślają raczej, polską terminologię. Negatywne skutki takiego stanu rzeczy dają się odczuć przede wszystkim przy szkoleniu nowych kadr.

Mimo wysokiego uznania dla opublikowanego w INFORMATYCE projektu słownictwa, uważam, że słowniczek ten wymaga pewnych poprawek, uzupełnień i dopracowania.

Pozwalam sobie przesłać: 1. Uwagi ogólne dotyczące projektu terminologii. 2. Propozycje zmian i uzupełnień poszczególnych haseł.

Dodam marginesowo, że w przypadku istnienia rozbieżności, jestem jak najdalszy od chęci forsowania terminologii ośrodka, z którym jestem związany.

W nadziei, że moje uwagi przydadzą się, pozostaję

z poważaniem

mgr Stefan Zawadzki

Uwagi ogólne do projektu terminologii

A. **Zasady redagowania słowniczka**

1. Słowniczek powinien być tak zredagowany, by mogli z niego korzystać także laicy, w tym również literaci, publicyści, dziennikarze i tzw. działacze. Projekt, zamieszczony w numerach 5, 6 i 7 „Informatyki" warunku tego niestety nie spełnia.

2. Wszędzie, gdzie tylko wchodzi to w grę, a szczególnie przy hasłach oznaczających czynności i działania maszyny, programisty lub operatora, powinny być podawane odsyłacze do hasła oznaczającego pojęcie przeciwne, a więc na przykład:

„język ukierunkowany maszynowo" — „język ukierunkowany problemowo",

„czytać" — „zapisywać",

„kodować" — „dekodować", itd.

3. Definicje haseł nie śmią pozostawiać wątpliwości co do nadrzędności, równorzędności (nie mylić z równoznacznością) i podrzędności pojęć.

4. Należy unikać przydawania nowych znaczeń wyrazom, będącym w powszechnym użyciu, tzn. należy unikać adaptowania do potrzeb informatyki takich pospolicie używanych wyrazów, które mają w polskiej mowie odmienne i od dawna zakorzenione znaczenie.

5. Przy hasłach oznaczających czynności i działania należy w słowniczku konsekwentnie stosować zasadę jednolitej formy gramatycznej: albo bezokolicznik (czytać, wyprowadzać itd.), albo rzeczownik odsłowny (czytanie, wyprowadzanie). Podawanie obydwu form uważam za zbędne, proponuję stosować jednolicie bezokolicznik.

B. **Ogólne zasady co do terminologii**

6. Jeżeli chodzi o „dane" i „informacje" to podzielam opinię dr Targowskiego zamieszczoną w książce „Informatyka — klucz do dobrobytu" na str. 107/8.

Skoro jednak mamy w naszej mowie polskiej te dwa różne wyrazy, to uważam, że można nadać im zróżnicowane znaczenia, przyjmując za kryterium punkt widzenia użytkownika. Oto trzy przykłady: I. „Panie Kowalski — mówi szef — prosiłem Pana o jedną konkretną informację, a pan zasypuje mnie mnóstwem bezużytecznych danych". II. Ta sama treść, zawarta np. w arkuszu spisowym, stanowiąca dla GUS tylko dane, mogłaby być dla MO lub Urzędu Skarbowego (gdyby nie tajemnica spisowa) cenną informacją.

III. Treść bilansu przedsiębiorstwa stanowi dla oddziału banku informację o stanie przedsiębiorstwa, a dla centrali banku są to fragmentaryczne dane służące do sporządzenia informacji o sytuacji w branży lub resorcie.

7. Trudno mi pogodzić się z terminem „kartoteka" w odniesieniu do treści zapisanych na nośniku magnetycznym, skoro nie ma tam ani kart, ani niczego, przypominającego tekę. Mimowoli przypomina się stary dowcip: dlaczego lombard nazywa się (po włosku) „monte di pietà" czyli „góra miłosierdzia" skoro nie ma tam w rzeczywistości ani góry, ani miłosierdzia.

W ośrodku, w którym pracuję, używa się — jako odpowiednika angielskiego „file" i niemieckiego „Datei" — nazwy ZBIÓR DANYCH albo krótko ZBIÓR. Nazwę tę stosuje w swoich skryptach i książkach również M. Szaniawska.

Zupełnie nieudany jest „plik". Przypomina on łacińsko-lekarską nazwę kołtuna — „plica polonica". Z uwagi na to, że z ETO będą korzystać w przyszłości także i lekarze, należałoby niefortunny „plik" obciąć i spalić.

8. Nie znajduję w słowniczku niezbyt wygodnego — moim zdaniem — określenia „jednostka przetwarzarzania", brak jednakże też innego określenia na TO, co z punktu widzenia użytkownika wykonuje się na komputerze.

Proponuję wyraz ZADANIE (Czesi mówią „úloha" co znaczy właśnie „zadanie"). Przy czym ZADANIA mogłyby być OKRESOWE (np. księgowość, statystyka), wykonywane regularnie codziennie, co dekadę czy choćby raz do roku, lub DORAŹNE (np. obliczenie konstrukcji mostu), wykonywane w miarę potrzeby.

9. Nie do przyjęcia, zarówno moim zdaniem, jak i zdaniem wielu moich kolegów, jest tłumaczenie angielskiego „statement" jako „instrukcji".

Po pierwsze, gdyby chcieć wiernie przetłumaczyć, to wyraz „statement" znaczy „stwierdzenie, oświadczenie".

Po drugie, polski wyraz „instrukcja" jest całkowicie jednoznacznie „ocechowany znaczeniowo" i oznacza (wg słownika wyrazów obcych) „przepis postępowania" lub „zbiór przepisów". Ponieważ mamy wszyscy (zarówno informatycy, jak i profani) stale do czynienia z tyloma różnymi instrukcjami (od instrukcji dołączonej do maszynki do golenia po instrukcję kredytową), przeto przydanie temu pospolitemu wyrazowi jeszcze jednego znaczenia wprowadziłoby tylko zamęt.

Proponuję tłumaczyć „statement" jako WYRAZ lub WYRAŻENIE. Wprawdzie także nie jest to tłumaczenie dosłowne, ale zdaje mi się najbliższe tego, o co chodzi.

10. Brakuje haseł i wyjaśnień dotyczących różnych typów ZBIORÓW („kartotek", „plików"), jak zbiór: testowy, wejściowy, wyjściowy, wynikowy (vide załącznik 2).

11. Należy bezwarunkowo inaczej i dokładniej zdefiniować różne typy programów (część II, l.p. 56—71) aby nie było wątpliwości, które pojęcie jest nadrzędne a które podrzędne. Nie wszystkie rodzaje

programów zostały w słowniczku wymienione i zdefiniowane. Braki pozwalam sobie uzupełnić w zał. 2. (Program: użytkowy, pokazowy, systemowy, operatorski).

12. Definicje różnych rodzajów języków (część II, l.p. 29—37) są we wzajemnym kontekście niezrozumiałe. Proponuję odmienną redakcję tych haseł (załącznik 2).

13. Sądzę, że można z powodzeniem usunąć ze słowniczka niektóre hasła. Należą do nich: algorytm, — gdyż jest to pojęcie z zakresu liceum ogólnokształcącego, a następnie „adresowanie bezwzględne, pośrednie, symboliczne i względne", gdyż jeśli zdefiniujemy różne rodzaje adresów i pojęcie adresowania w ogóle, to oczywiste się staje znaczenie i tamtych haseł.

14. Jeżeli angielski wyraz „to debug" (dosłownie „odpluskwiać") zechcemy tłumaczyć jako „uruchamiać program", to w takim razie jak nazwiemy TO, co robi się, aby komputer pracował? Komputer uruchamia się nie przez samo włączenie do sieci, ale przez uruchomienie w nim jakiegoś programu. Natomiast testowanie czyli sprawdzanie programu nie jest sztuką dla sztuki, ale ma na celu oczyszczenie programu z błędów, czyli odpluskwienie go. Proponuję więc, by angielskie „to debug" również tłumaczyć jako TESTOWAĆ lub SPRAWDZAĆ.

PROPOZYCJE ZMIAN I UZUPEŁNIEŃ DO PROJEKTU TERMINOLOGII E.T.O.

Do części I: DANE (nr 5/71)

L.p. 3 ANALOGOWY, rubryka 3.
Powinno być chyba „zmieniających się" a nie „zmniejszających". Ponadto proponuję dodać na końcu: „Przeciwieństwo — patrz l.p. 25, DYSKRETNY"

L.p. 4 rubryka 2.
Proponuję dodać trzecie równoznaczne hasło „AUTOMATYCZNE PRZETWARZANIE INFORMACJI (API)"

L.p. 5 BAJT, rubryka 3.
Bajt nie jest umowną liczbą bitów, ale całkiem ściśle ustaloną, i nie najczęściej, ale zawsze 8. Proponuję zmienić tylko pierwsze zdanie tego hasła jak następuje: „Podstawowa komórka pamięci w większości komputerów nowszego typu, składająca się z 8 bitów".
Dalej bez zmian.

L.p. 9 CZYTAĆ, rubryka 3.
Zamiast dotychczasowego — proponuję tekst następujący: „przenosić dane z nośników danych lub z pamięci zewnętrznej do pamięci wewnętrznej (operacyjnej). Działanie odwrotne — ZAPISYWAĆ".

L.p. 19 DEKODOWAĆ, rubr. 3.
Proponuję dodać na końcu: „Działanie przeciwne — KODOWAĆ".

L.p. 21 DŁUGOŚĆ SŁOWA, rubr. 3.
Proponuję inny tekst:
„Liczba oznaczająca rozmiary jednego słowa, której mianem w zależności od typu komputera są bity, bajty, znaki lub sylaby".

L.p. 25 DYSKRETNY rubr. 3.
Proponuję dodać na końcu: „Przeciwieństwo — patrz ANALOGOWY".

L.p. 27 INFORMACJA, rubr. 3.
Proponuję definicję, która wydaje mi się jaśniejsza i dokładniejsza:
„Dane, które są użyteczne dla ich odbiorcy lub użytkownika".

L.p. 34 KODOWAĆ, rubr. 3.
Proponuję dodać na końcu: „Działanie przeciwne — DEKODOWAĆ".

L.p. 42 PAKOWAĆ, rubr. 3.
Proponuję dodać na końcu słowa:
„Działanie przeciwne — ROZPAKOWAĆ".

L.p. 43 rubryka 2.
Jako polski odpowiednik wyrazu „file" proponuję mimo wszystko określenie „ZBIÓR DANYCH" lub krótko „ZBIÓR".

L.p. 44 rubr. 2.
Analogicznie „master file" — ZBIÓR GŁÓWNY".

L.p. 45 rubr. 2.
„Transaction file" proponuję tłumaczyć:
ZBIÓR TRANSAKCJI i/lub ZBIÓR OBROTÓW.

L.p. 55 rubryka 3.
Proponuję dodać na końcu słowa:
„Działanie przeciwne — PAKOWAĆ".

L.p. 56 SŁOWO, rubryka 3.
Proponuję treść odmienną, jak następuje:
„1) w niektórych komputerach, zwłaszcza starszego typu, podstawowa komórka pamięci, zawierająca od 24 do 42 bitów, w zależności od typu komputera,
2) w komputerach, w których podstawową komórką pamięci jest znak, sylaba lub bajt — określona liczba znaków, bajtów lub sylab (np. w NCR-315 od 1 do 8 sylab), którą dla celów przetwarzania można wzgl. trzeba użyć jako wydzieloną jednostkę danych".

L.p. 58 SYLABA, rubr. 3.
Proponuję treść odmienną, a mianowicie:
„Podstawowa komórka pamięci w komputerze NCR-315, złożona z 12 bitów. W sylabie można umieścić 2 znaki alfanumeryczne albo 3 znaki numeryczne".

L.p. 65 rubryka 2.
Proponuję przyjęcie wszędzie jednakowej formy gramatycznej, tj. czasownika w bezokoliczniku:
„WPROWADZAĆ DANE".
W rubr. 3 proponuję dodać na końcu:
„Działanie przeciwne — WYPROWADZAĆ DANE".

L.p. 68 Proponuję analogiczne zmiany:
Rubr. 2 „WYPROWADZAĆ DANE",
Rubr. 3 „Działanie przeciwne — WPROWADZAĆ DANE".

L.p. 75 ZAPISYWAĆ, rubr. 3.
Proponuję treść następującą:
„Umieszczać dane w pamięci zewnętrznej komputera lub na nośniku danych. Działanie przeciwne — CZYTAĆ".

L.p. 6 rubryka 3.
Ze względów, o których była mowa w załączniku 1, proponuję nie „plików", lecz „zbiorów".

L.p. 12 rubryka 3.
Proponuję dodać na końcu takie zdanie:
„W niektórych maszynach do znaków cyfrowych zalicza się też niektóre znaki specjalne, np. w systemie NCR-315 także a , ⊞ & . — (akumulator, przecinek, spację, ampersand, kropkę i minus).

L.p. 17a rubryka 2: DANE TESTOWE
rubryka 3: Dane, niekoniecznie prawdziwe, lecz rozmyślnie tak spreparowane, by zawierały wszelkiego rodzaju błędy, i by mogły dzięki temu służyć do → testowania programów.

L.p. 45a rubryka 2: ZBIÓR TESTOWY
rubryka 3: Zbiór, zawierający → dane testowe.

L.p. 45b rubryka 2: ZBIÓR WEJŚCIOWY (source file)
rubryka 3: Zbiór zawierający → dane wejściowe.

L.p. 45c rubryka 2: ZBIÓR WYJŚCIOWY (destination file)
rubryka 3: Zbiór zawierający → dane wyjściowe.

L.p. 45d rubryka 2: ZBIÓR WYNIKOWY
rubryka 3: Zbiór wyjściowy zawierający → informacje.

Do części II — PROGRAMOWANIE (nr 6/71)

L.p. 8 Adres rozkazu — hasło zbędne, znaczenie oczywiste.

L.p. 11 rubryka 2: nie „Adresowanie" lecz „ADRESOWAĆ".

L.p. 12 — 15 Hasła zbędne, znaczenie oczywiste.

L.p. 16 Algorytm — hasło zbędne.

L.p. 22 DEKLARACJA
rubryka 3: proponuję nieco inne określenie:
„Wyrażenie języka programowania, określające typy zmiennych, obszary w pamięci lub stałe programu. Inaczej PSEUDOROZKAZ".

L.p. 24 DYREKTYWA
rubryka 3:
„Wyrażenie mające postać rozkazu języka programowania, lecz służące do sterowania tłumaczeniem programu".

L.p. 28 rubryka 2: WYRAŻENIE (a nie „instrukcja")
rubryka 3:
„Deklaracja, rozkaz, pseudorozkaz lub makrorozkaz języka programowania".

L.p. 29 JĘZYK WEWNĘTRZNY, JĘZYK MASZYNY
Kod, złożony z symboli odpowiadających wyłącznie rozkazom maszyny i zapisywanych jako różne kombinacje znaków cyfrowych. Przeciwieństwo: język programowania.

L.p. 30 JĘZYK PROGRAMOWANIA
Sztuczny język przystosowany do pisania programów dla komputerów. Wyrazy i wyrażenia j.p. są to bądź wyrazy języka naturalnego czyli etnicznego (angielskiego, polskiego, niemieckiego itp.) w pełnym brzmieniu, bądź — przeważnie — skrócone i uproszczone, bądź wreszcie wyrazy sztuczne lecz o brzmieniu łatwym do zapamiętania. J.p. musi być przy pomocy programu tłumaczącego lub kompilującego przetłumaczony na język wewnętrzny maszyny.

L.p. 31 JĘZYK SYMBOLICZNY
Najprostszy język programowania, ukierunkowany maszynowo, złożony wyłącznie z wyrazów odpowiadających rozkazom maszyny.

L.p. 32 JĘZYK UKIERUNKOWANY MASZYNOWO
Język programowania przystosowany do pisania programów tylko dla określonego typu komputerów (np. NEAT — dla NCR-315, NEAT/3 — dla NCR-615, PLAN — dla ICT i „Odra" itp.). Przeciwieństwo: Język niezależny od maszyny.

L.p. 33 JĘZYK NIEZALEŻNY OD MASZYNY
Język programowania o charakterze uniwersalnym, przeznaczony do pisania programów dla różnych marek i różnych typów komputerów, pod warunkiem, że będą one wyposażone w odpowiednie translatory. Przeciwieństwo: Język ukierunkowany maszynowo.

L.p. 34 Język źródłowy
Język, w którym został napisany program. Jeżeli j.ź. nie jest językiem wewnętrznym maszyny, to dany program musi być przy pomocy odpowiedniego translatora przetłumaczony na język wewnętrzny.

(Uwaga: hasła „Język algorytmiczny", „Język problemowy" i „Język proceduralny" proponuję pozostawić bez zmian.)

L.p. 38 rubryka 3, proponuję zamieścić wyłącznie odsyłacz: Patrz „język maszyny".

L.p. 39 „Kod rozkazów" — hasło zbędne.

L.p. 41 „Lista rozkazów" — hasło zbędne, znaczenie oczywiste.

L.p. 43 MAKROROZKAZ
rubryka 3, proponuję tekst: „Wyraz języka programowania, przeznaczony do realizacji złożonej operacji za pomocą sekwencji rozkazów maszyny. Makrorozkaz po przetłumaczeniu przez kompilator rozwija się w rodzaj niewielkiego podprogramu".

L.p. 19 BIBLIOTEKA PROGRAMÓW
Proponuję nieco inną definicję, a mianowicie:
„Zbiór sprawdzonych programów zaopatrzonych w opisy umożliwiające łatwe korzystanie z nich. Rozróżnia się bibliotekę → programów systemowych i bibliotekę programów → użytkowych".

L.p. 28a INICJALIZACJA
Przygotowanie do użytku → zestawu kart magnetycznych. taśmy magnetycznej lub pakietu dysków, polegające na a) dokonaniu kontrolnego zapisu i odczytu w celu stwierdzenia, czy nie ma przypadkiem nieczytelnych bloków, b) wyznaczeniu — jeżeli chodzi o karty magnetyczne — ścieżek zastępczych, c) ocechowaniu danego nośnika przez zapisanie na początku taśmy magnetycznej odpowiedniego numeru lub symbolu zaś na początku każdej ścieżki na kartach magnetycznych numeru zestawu, numeru karty i numeru ścieżki.

L.p. 50a PĘTLA NIESKOŃCZONA
Sekwencja rozkazów, które wskutek błędu programisty lub defektu komputera wykonują się bez przerwy aż do zatrzymania maszyny.

L.p. 55a PROGRAMY SYSTEMOWE
Programy dostarczane przez producenta wraz z maszyną, bez których niemożliwe jest funkcjonowanie komputera; należą tu programy systemu operacyjnego, generatory, kompilatory, inicjalizatory, częściowo programy usługowe itp. Przeciwieństwo: programy użytkowe.

L.p. 55b PROGRAMY OPERATORSKIE
Programy usługowe (pomocnicze), których uruchomienie leży w gestii obsługi (operatorów) komputera.

L.p. 55c PROGRAMY UŻYTKOWE
1) Programy wchodzące w skład → zadania, 2) Programy pojedyncze, przetwarzające bezpośrednio → dane wejściowe na → informacje. Rozróżnia się p.u. UNIWERSALNE, mogące służyć wielu użytkownikom oraz p.u. SPECJALNE, służące do przetwarzania danych jednego określonego użytkownika.

L.p. 56 „Program biblioteczny" — hasło zbędne, gdyż każdy sprawdzony program wchodzi w skład jakiejś biblioteki programów.

L.p. 57 rubryka 3, proponuję skreślić słowa „lub błędów w programie", gdyż do tego służą programy testujące.

L.p. 58 rubryka 3: Program systemowy...

L.p. 59 PROGRAM GENERUJĄCY
Proponuję inne określenie: „Program systemowy, który za pomocą jedynie parametrów ustalonych przez programistę generuje program użytkowy w postaci programu wynikowego".

L.p. 60 rubryka 3: „Program systemowy tłumaczący" itd.

L.p. 61 PROGRAM KOMPILUJĄCY
Proponuję inne określenie: „Program systemowy przeznaczony do przekształcania programów napisanych w języku programowania na programy wynikowe, z jednoczesnym wykrywaniem błędów formalnych, ortograficznych".

L.p. 62 „Program maszyny" — hasło zbędne. Jest to po prostu — program wynikowy.

L.p. 63 PROGRAM POST MORTEM
rubryka 2 — dopisać „memory dump"
rubryka 3 — proponuję inne określenie:

„Program systemowy pomocniczy umożliwiający wypisanie w formie tabulogramu zawartości całej lub niektórych obszarów pamięci operacyjnej".

L.p. 64 PROGRAM SORTOWANIA
rubryka 3: „Program użytkowy.." itd.

L.p. 65 rubryka 3: „Program systemowy.." itd.

L.p. 68 TRANSLATOR
rubryka 3: proponuję odmienne określenie:
„Program systemowy przeznaczony do przekształcania programów źródłowych napisanych w języku symbolicznym na → programy wynikowe".

L.p. 71 PROGRAM INICJALIZUJĄCY
INICJALIZATOR
initializer program
Program systemowy służący do → inicjalizacji taśm magnetycznych, zestawów kart magnetycznych lub pakietów dysków.

L.p. 81 REDAGOWAĆ
Proponuję odmienne określenie:
„Przekształcać zawartość zapisu (recordu) do takiej postaci, jaka powinna znaleźć się w tabulogramie".

L.p. 91 rubryka 2.
SEGMENT (segment, overlay)
rubryka 3: Wydzielona część programu mieszcząca się jednorazowo w pamięci operacyjnej komputera.

L.p. 104 rubryka 2.
TESTOWAĆ (SPRAWDZAĆ)
to test, to debug.
rubryka 3
Uruchamiać prawidłowo skompilowany program przy pomocy —— danych testowych w celu wykrycia ewentualnych błędów logicznych.

L.p. 106 rubryka 2: URUCHAMIAĆ PROGRAM
rubryka 3: spowodować wczytanie do pamięci operacyjnej programu i wykonywanie się kolejnych rozkazów.

L.p. 110 ZADANIE
Przetwarzanie danych na komputerze charakteryzujące się tym, że jego rozmiar i jakość (rodzaj działań, operacji, obliczeń) stanowią całość zaspokajającą wymagania użytkownika co do rozwiązania określonego problemu ekonomicznego lub technicznego. ZADANIE może składać się z jednego lub z większej liczby programów i rozpoczyna się w zasadzie od wprowadzenia do komputera → danych wejściowych, a kończy się wyprowadzeniem → informacji w postaci wymaganej przez użytkownika, a więc w postaci tabulogramu lub prezentacji na ekranie.
ZADANIE OKRESOWE — odbywające się regularnie w określonych terminach.
ZADANIE DORAŹNE — odbywające się w miarę potrzeby.
ZADANIE WIELOWARIANTOWE
● zadanie wielowariantowe doraźne — np. system rezerwacji biletów lotniczych, który odpowiada na różne zapytania (to do Honolulu, to do Berdyczowa, jednym razem na dziś, innym razem na miesiąc naprzód).

● zadanie wielowariantowe okresowe — np. księgowość bankowa, wykonywana codziennie, z tym że w określonych terminach uruchamia się dodatkowe programy obliczające odsetki, dopisujące je do rachunków, przeksięgowujące kredyty itp.

Zadania mogą łączyć się w SYSTEMY INFORMACYJNE, jednakże w niewielkiej instytucji system informacyjny może składać się nawet tylko z jednego zadania.

Do części III — SPRZĘT (nr 7/71)

L.p. 9 rubryka 2.
Nie „Czytnik/dziurkarka" ale CZYTNIK-DZIURKARKA

L.p. 10 analogicznie powinno być: CZYTNIK-SORTER

L.p. 5a CRAM, Card Random Access Memory
Pamięć zewnętrzna o dostępie bezpośrednim na kartach magnetycznych.

L.p. 51a PAKIET CRAM
ZESTAW CRAM
(deck)
Wymienny komplet kart magnetycznych. Liczba kart w pakiecie (zestawie) jest zawsze stała dla określonego typu komputera (np. NCR-315 — 256 kart).

L.p. 87a ZASOBNIK CRAM
Pudełko z tworzywa sztucznego służące do przechowywania pakietu (zestawu) kart magnetycznych i zaopatrzone w specjalne trzpienie metalowe umożliwiające łatwe i szybkie wkładanie pakietu do urządzenia CRAM i wyjmowanie.

L.p. 32 rubryka 3.
Po słowach „Elementarna jednostka pamięci" proponuję dodać: „którą może być, zależnie od typu komputera, słowo, sylaba lub bajt".

Drodzy Czytelnicy!

Czekamy na Wasze listy do nowego działu

TRYBUNA CZYTELNIKA

Przysyłajcie do naszej redakcji wypowiedzi na wszelkie interesujące i nurtujące Was sprawy z dziedziny informatyki!

Komu komputer?

W listopadzie 1971 r. w Krajowym Biurze Informatyki obradowano nad planami instalacji komputerów w najbliższych latach, nad wnioskami z narady Zespołów Koordynacji Terenowej Rozwoju Techniki Obliczeniowej oraz nad sprawami szkolenia kadr dla informatyki. Przedmiotem osobnej dyskusji były zasady funkcjonowania Krajowego Biura Informatyki. Naradzie przewodniczył zastępca Dyrektora Generalnego KBI — dr inż. Andrzej Targowski. Wśród zaproszonych znaleźli się przedstawiciele resortów i przemysłu sprzętu informatyki, a ze strony Komitetu Nauki i Techniki udział wziął zastępca Przewodniczącego KNiT — prof. dr Zbigniew Jasicki.

Analiza i porównanie łącznego zapotrzebowania resortów na poszczególne typy komputerów wykazuje, że zapotrzebowanie zgłaszane przez resorty na znane typy komputerów, takie jak ODRA-1304 czy ODRA-1305 znacznie przewyższa przewidywane rozmiary produkcji. Natomiast w przypadku komputerów, których seryjna produkcja będzie podejmowana w najbliższych latach, jak np. ODRA-1325, R-30 czy K-202 zapotrzebowanie jest znacznie niższe od możliwości produkcyjnych. Tłumaczyć to można zarówno niedostateczną informacją ze strony producenta na temat nowości produkcyjnych, jak i pewnym konserwatyzmem autorów programów resortowych.

W planie rozmieszczenia komputerów w ciągu bieżącej 5-latki przewiduje się pokrywanie niedoborów w poszczególnych typach komputerów znanych, komputerami nowocześniejszymi, wprowadzanymi dopiero do produkcji. Porównanie liczb dotyczących zapotrzebowania z przemysłowymi planami produkcji wykazuje, że począwszy od 1973 r. nastąpić powinno stopniowe pokrycie resortowych potrzeb na komputery. Jedynie jeszcze w 1972 r. około 30% potrzeb resortowych na komputery nie znajdzie pokrycia w produkcji krajowej. Należy pamiętać również o tym, że w ciągu najbliższych kilku lat będzie występowała konieczność wycofania kilkudziesięciu przestarzałych i wyeksploatowanych komputerów, zwłaszcza do obliczeń numerycznych.

Podstawowym założeniem przy opracowywaniu planu rozmieszczenia komputerów na lata 1971—1975 jest zabezpieczenie potrzeb sprzętowych realizatorów problemów i zadań węzłowych oraz szkolnictwa (dla celów dydaktycznych). To właśnie szkolnictwo w pierwszym rzędzie powinno przygotować do zawodu blisko 34 000 specjalistów. Oblicza się bowiem, że łączne zatrudnienie w informatyce na koniec bieżącej 5-latki zamknie się liczbą ca 55 000 pracowników. Oczywiście samo szkolnictwo wyższe i średnie nie podoła temu zadaniu. W kształceniu nowej kadry informatyków konieczna jest pomoc i współpraca wielu instytucji. Przygotowanie specjalistów od zastosowań i eksploatacji, tj. projektantów systemów, programistów, operatorów i służby konserwatorskiej to tylko jedna strona zagadnienia. Konieczne jest również szerokie szkolenie kadr użytkowników, którego to zadania podjęła się m. in. Telewizja, prowadząca od lat kursy politechniczne.

Funkcje generalnego koordynatora szkolenia i doskonalenia kadr informatyki w skali krajowej podjął Ośrodek Badawczo-Rozwojowy Informatyki (OBRI). Uczestnicy dyskusji przyjęli następujące wnioski do realizacji:

- należy polepszyć tryb dostaw sprzętu drogą wcześniejszego składania zamówień do producentów, przy czym ze względu na cykl produkcji komputera, powinno to być wyprzedzenie przynajmniej roczne, zakładając możliwość pełnego pokrycia krajowych potrzeb na komputery począwszy od 1973 r. oraz pełne przygotowanie ELWRO do świadczenia kompleksowych usług na rzecz użytkowników, należy rozważyć możliwość wprowadzenia systemu dzierżawy sprzętu, konieczne jest przystąpienie do prac nad zmianą taryfikatora kwalifikacyjnego pracowników informatyki, a więc nad uporządkowaniem listy stanowisk i wymaganych kwalifikacji oraz związanego z tym poziomu płac,
- należy opracować system dostaw sprzętu i materiałów eksploatacyjnych zarówno produkcji krajowej, jak i z importu,
- należy wprowadzić obowiązek wyprzedzających informacji o parametrach sprzętu informatyki, którego produkcję się uruchamia (np. R-30).

Przegląd prasy krajowej

„GOLEM" sprzedawany jest dopiero od roku, ale już kupiono 22 systemy. Podczas europejskiej konferencji „DIEBOLDA" w Paryżu, na której omawiano ostatniej jesieni konkretne przykłady postępu w informatyce w ciągu ubiegłych trzech lat, jednym z najciekawszych wykładów było wystąpienie prof. H. V. Woltersa. Jest on doradcą naukowym koncernu Siemensa i omawiał swoje dzieło — właśnie system GOLEM. Ta uniwersalna pamięć współpracuje z elektroniczną maszyną cyfrową SIEMENS 4004. Rozmawia się z nią językiem naturalnym. I to pozwala na dość powszechne stosowanie elektronicznej techniki obliczeniowej w dziedzinach, do których komputery miały dotychczas utrudniony dostęp. Tak więc Wydział Filozofii w Düsseldorfie, wespół z Uniwersytetem w Leningradzie, zajmuje się przy pomocy GOLEMA analizą języka Immanuela Kanta. Podobny system pamięci zastosowano w dziale prac badawczych Volkswagena, w koncernie Bayera, w kilku ministerstwach, w Towarzystwie Matematycznym, a także w Federalnym Urzędzie Prasowym NRF, dzięki czemu korespondent „ŻYCIA WARSZAWY" mógł z Paryża uzyskać informację, kiedy ukazał się np. komentarz polskiego dziennikarza na temat układu Polska-NRF, i na kilka innych, na miejscu sformułowanych pytań. Wrażenia z rozmowy z komputerem opisał Karol Szyndzielorz w „ŻYCIU i NOWOCZESNOŚCI" nr 300/71.

ZEWNĘTRZNIE „SEIKO MARU" nie różni się od innych zbiornikowców, ale jego załoga podczas rejsów korzysta z niezwykłego konsultanta, którym jest wieloczynnościowy komputer morski. W dwóch pierwszych podróżach, trwających po 940 godzin, pracę komputera obserwował przedstawiciel Control System Engineering Dept. Ishikawajima — Harima Heavy Industries Ltd., p. Nuzomi Sakano, który opublikował swe spostrzeżenia, co posłużyło „TYGODNIKOWI MORSKIEMU" nr 49/71 do opracowania futurologicznego artykułu o perspektywach komputeryzacji żeglugi morskiej.

EKSPERT OD ROBOTÓW, Charles A. Rosen ze Stanford Research Institute twierdzi, że nadejście inteligentnych maszyn w przemyśle jest rzeczą pewną. Już obecnie pracuje się nad skonstruowaniem komputera, który by reagował na 1000 słów, niezależnie od tego, kto je wypowie. Szef programu — Stephen Crocker spodziewa się, że komputer taki uda się skonstruować w ciągu 5 lat. Badacze są przekonani, że „inteligentne maszyny" mogą dokonać takiego postępu w automatyce przemysłowej, że uda się np. stworzyć możliwość masowej produkcji „ręcznie szytej" odzieży na zamówienie dzięki sy-

stemowi, który będzie znał wymiary każdego z nabywców. Artykuł o aktualnie prowadzonych na świecie badaniach nad „myślącymi" maszynami zamieścił amerykański tygodnik BUSINESS WEEK, a przedrukował go „TYGODNIK KULTURALNY" nr 49/71.

O POWSTANIU MIĘDZYNARODOWEGO KLUBU użytkowników maszyn cyfrowych ODRA pisze „SŁOWO POLSKIE" nr 287/71. Jednym z głównych partnerów polskiego przemysłu komputerowego jest Czechosłowacja; w jej ośrodkach obliczeniowych pracuje już 45 maszyn cyfrowych produkcji ELWRO i tam też powstał pierwszy tego rodzaju klub, a ponieważ w NRD pracuje 36 wrocławskich maszyn, na Węgrzech 16, nie mówiąc już o Związku Radzieckim, gdzie zainstalowano ich najwięcej — na poświęconej ODRZE międzynarodowej konferencji w Szpindlerowym Młynie — postanowiono utworzyć podobne kluby w pozostałych krajach-użytkownikach naszych komputerów i, oczywiście, związek tych klubów w postaci klubu międzynarodowego.

CO DALEJ Z NAUKĄ? Jerzy Surdykowski w nr 49/71 „PERSPEKTYW" omawia wyniki ankiety, którą w 1963 roku rozpisała amerykańska RAND-Corporation pośród dużej grupy wybitnych uczonych różnych krajów. Ankieta polegała na tym, iż każdy z uczestników podawał najbardziej prawdopodobny okres czasu lub nawet rok, w którym zostaną dokonane albo wdrożone do praktyki przyszłe odkrycia, jakie umyślili sobie autorzy. Wielu uczestników ankiety przewidywało już na lata 1970—71 wdrożenie takich osiągnięć cywilizacyjnych, jak: opłacalnego ekonomicznie odsalania wody morskiej, taniej, prostej i skutecznej kontroli urodzeń, nowych, ultralekkich tworzyw budowlanych, maszyn do automatycznego tłumaczenia, rozpowszechnienia się przeszczepów i protez narządów ludzkich. Żadne z tych domniemanych osiągnięć nie stało się rzeczywistością cywilizacyjną. Nie znaczy to, oczywiście, że przewidywane przez ankieterów wydarzenia nie nastąpią. Oznacza to tylko, że następują one wolniej, niż przypuszczano. Autor artykułu przytacza tu tezę amerykańskiego naukoznawcy — Dereka J. de Solla Price, który przewiduje przekształcenie się obecnego, szybkiego rozwoju typu wykładniczego w nowy, znacznie powolniejszy, odbywający się według krzywej logistycznej. W układzie ograniczonym, składającym się z ograniczonej liczby ludzi, środków finansowych i innych, nauka i wraz z nią technika nie mogą rozwijać się dowolnie długo w sposób nieograniczony. Zahamowanie musi nastąpić zarówno z powodu ograniczenia środków, jak i z powodu wewnętrznego „tłumienia adaptacyjnego".

49 NUMER „FORUM" przynosi tłumaczenie artykułu zamieszczonego w „DIE WELTWOCHE", a dotyczącego trudności, w jakie popadła angielska ICL i cały przemysł komputerowy i elektroniczny tego kraju. ICL jest największym pozaamerykańskim producentem komputerów, a Wielka Brytania jedynym rynkiem europejskim, którego nie zdołał opanować gigantyczny IBM. Poza tym przemysł budowy komputerów stanowi tę część technologii, którą wyspiarze pragną wnieść do małżeństwa z Europejską Wspólnotą Gospodarczą.

„CZY IMPAS USŁUG KOMPUTEROWYCH?" — pytanie to zadał „PRZEGLĄD TECHNICZNY" (nr 49/71) czołowym specjalistom informatyki polskiej. Chodzi nie tylko o rozwój produkcji komputerów w ramach międzynarodowego podziału pracy RWPG, ale także o ich zastosowanie zgodnie z zasadą najwyższej efektywności. Biorący udział w dyskusji Dyrektor Krajowego Biura Informatyki powiedział m.in.: „W dobie ilościowego rozwoju ośrodków obliczeniowych trzeba przyjąć zasadę, że doświadczone placówki ETO powinny stać się propagatorami informatyki. Twierdzę, że wszystkie ośrodki obliczeniowe powinny świadczyć usługi komercjalne w jakimś tam zakresie. Dla ZETO zasadniczym zadaniem perspektywicznym staje się świadczenie na rzecz administracji państwowej i terenowej. Drugim kierunkiem działania jest konsultacja i uruchamianie systemów informatycznych w przedsiębiorstwach przemysłowych, które rozpoczynają korzystanie z „ETO".

WOJEWÓDZTWO BIAŁOSTOCKIE przystępuje do komputeryzacji z pewnym opóźnieniem, ale już ma się czym pochwalić — donosi „GAZETA BIAŁOSTOCKA" nr 332/71. Trzy lata temu został tam zorganizowany ośrodek ZETO, który dysponuje trzema zestawami maszyn analitycznych i zatrudnia 95 osób, pracowników o wysokich kwalifikacjach. W roku 1973 rozpocznie się budowa nowoczesnego Centrum Obliczeniowego, którego komputery będą świadczyć usługi wielu przedsiębiorstwom przemysłowym województwa.

NIE BĘDZIE INŻYNIERA CZY EKONOMISTY opuszczającego uczelnię, który nie umiałby posługiwać się maszyną elektroniczną i nie widział możliwości, jakie ta technika stwarza. „TRYBUNA LUDU" nr 347/71 omawia problemy komputeryzacji regionu krakowskiego. Szersze wykorzystanie komputerów w regionie wymaga wyszkolenia kadry, która będzie je programować, obsługiwać i konserwować. Specjalistów, których corocznie dostarczają Wydział Matematyki UJ, Akademia Górniczo-Hutnicza i Wyższa Szkoła Ekonomiczna, jest ciągle za mało.

„ETAT TO 46 GODZIN TYGODNIOWO PRACY. Każdy z nas, etatowców dostaje kartę programową, do której wpisuje robione przez siebie tematy i liczbę godzin na nie przeznaczoną. Obie liczby (etatowa i programowa) muszą się zgadzać. Nie może zabraknąć godzin na programy, bo trzeba by było zwolnić pracownika. Jeśli np. temat jest łatwy i na jego realizację wystarczyłoby dwieście godzin — wpisujemy 450 godzin. A gdyby się wpisało 600 — komisja też by zatwierdziła. A przecież to fachowcy i orientują się, że to lipa. Dlaczego to robią? Nie mają po prostu innego wyjścia, jeśli biuro ma utrzymać etaty. Dwa tysiące trzysta godzin w roku musi być zapisane w każdej karcie. Niejednokrotnie więc, gdy nie ma tematów zleconych, wymyślamy sami takie, które mogą przydać się w życiu. A to nie takie proste. Ja sam jestem przekonany, że program, nad którym siedziałem pół roku, chyba nikomu się na nic nie przyda. To nie podnosi na duchu. Niech mi pani wierzy!"

A oto pytanie dziennikarki „KURIERA POLSKIEGO" (nr 296/71), która tych anonimowych zwierzeń jednego z pracowników ośrodka obliczeniowego wysłuchała: „Po co w takim razie zorganizowano ośrodek skoro spece od ekonomii udowodniliby zaraz, że utrzymywanie maszyny nie ma podstaw ekonomicznych?"

Odpowiedź bohatera reportażu: „Żeby unowocześnić gospodarkę"...

Opracowała ELK

Współpraca z zagranicą

FRANCJA

W okresie od końca kwietnia do początku listopada 1971 r. dwuosobowa delegacja pracowników Zakładu Elektronicznej Techniki Obliczeniowej ZOWAR w Warszawie odbyła 28-tygodniowe szkolenie w IBM Education Centre w Paryżu. Szkolenie w Paryżu było związane z kontraktem na zakup komputera IBM 360/M50 i dotyczyło hardware'u komputera. W ramach szkolenia w zakresie potrzebnym do konserwacji i usuwania uszkodzeń maszyny grupa uczestniczyła w następujących kursach:

Jednostka centralna i kanały IBM 360/M30 — b. ogólnie i pobieżnie
Jednostka centralna i kanały IBM 360/M40 — jako wstęp do M50
Jednostka centralna i kanały IBM 360/M50 — szczegółowo
Jednostka sterująca 2821
Jednostka sterująca 2803 Model i Model 2
Jednostka sterująca i stacje dysków 2314
Jednostka sterująca i stacje taśm 2415 Modele 4—6
Stacje taśm 2401/2403 Modele 1—6
Czytniko-dziurkarka kart 1402
Drukarka 1403
Monitor 1052

Praca w ośrodku szkoleniowym odbywa się w rytmie: cztery godziny zajęć teoretycznych i cztery godziny zajęć laboratoryjnych w każdym dniu roboczym.

Zajęcia teoretyczne prowadzone były bądź metodą tradycyjną, bądź metodą nauki z pomocą komputera, zwaną w języku IBM „FIS". Słuchacz kursu

uczący się metodą „FIS" otrzymuje do dyspozycji urządzenie końcowe podłączone do komputera pełniącego rolę nauczyciela oraz wszelkie pomoce naukowe w postaci skryptów, przezroczy, taśm magnetycznych.

Kursy dla ośrodka szkoleniowego w Paryżu prowadzi się z pomocą komputera IBM 360/M50, który jest zlokalizowany w Londynie. W ten sposób prowadzi się również kursy dla innych europejskich ośrodków szkoleniowych.

Uczestnicy kursu podkreślają wyższość metody FIS nad metodą tradycyjną: nauka wymaga dużej i nieprzerwanej koncentracji i jest przez to wydajniejsza, jak również eliminuje się ewentualne niedoskonałości tradycyjnego wykładowcy.

Wszystkie kursy, jakie odbyła grupa ZOWAR-u zyskały wysoką ocenę uczestników zarówno od strony merytorycznej jak i pod względem metodycznym.

J. Chorzewski

BUŁGARIA

Wyjazd polskiej delegacji do Bułgarii (w listopadzie zeszłego roku) nastąpił w uszczuplonym składzie. Mimo wyraźnie sprecyzowanego celu tego polsko-bułgarskiego spotkania — ETO w hutnictwie obu krajów — wyjechali tylko dwaj przedstawiciele Ośrodka Badawczo-Rozwojowego Informatyki. Zabrakło natomiast strony, zdawałoby się najbardziej zainteresowanej; kierownictwo hutnictwa polskiego odmówiło delegowania swoich przedstawicieli. Świadczy to o zupełnym braku rozeznania przez stronę polską w obecnym stanie rozwoju informatyki bułgarskiej i planach na przyszłość strony bułgarskiej, szczególnie w branży hutniczej.

Bez wątpienia hutnictwo bułgarskie — 2 huty produkujące obecnie około 2 milionów ton stali — jest niewielkie w porównaniu z 26 hutami polskimi produkującymi 11 milionów ton stali. Inne w związku z tym są problemy, inna skala trudności i inne środki, jakie należy zaangażować. Obserwowany w Bułgarii rozmach, bardzo szeroki front prowadzonych robót, zaangażowanie wielu komórek na różnych szczeblach, w kraju i za granicą, wskazuje na wagę, jaką bułgarskie władze hutnictwa przywiązują do informatyki. Wskazuje na to również taki szczegół, jak przyznanie etatu dyrektora dla kierownika centrum EZO Kombinatu Hutniczego w Kremikowcach. Rozmówcy nasi w CNIPIA nie ukrywali trudności i kłopotów z wdrażaniem systemów, a wynikających z nieodpowiedniego stosunku niektórych pracowników przemysłu do zagadnień informatyki. Skierowanie głównego nurtu prac na automatyzację zbierania i przekazywania danych stworzy źródło informacji bardzo dokładnych, obszernych a jednocześnie omijających tradycyjne dokumenty źródłowe. W oparciu o taką bazę — dane odpowiednio przygotowane będą stanowiły prawidłowy fundament pod prace w zakresie przetwarzania. Pozwoli to stronie bułgarskiej uniknąć, w okresie wdrażania nawet najlepszych systemów, niepowodzeń związanych z brakiem poprawnych danych.

Organizacja informatyki w Bułgarii

Najwyższym organem decydującym o zasadniczych sprawach związanych z informatyką w LRB jest Gospodarczy Komitet Nauki i Techniki. Wyznacza on kierunki rozwoju informatyki, kontroluje osiągane wyniki.

Ustalanie odpowiednich planów i czuwanie nad przebiegiem ich realizacji należy do właściwych komórek wchodzących w skład:

Ministerstwa Przemysłu Maszynowego

Ministerstwa Przemysłu Ciężkiego

Ministerstwu Przemysłu Maszynowego podlegają sprawy związane z wyposażeniem kraju w potrzebną bazę techniczną i rozwojem informatyki w podległych zakładach. Podległy Ministerstwu Przemysłu Maszynowego IZOT jest zasadniczym organem, który właściwie kieruje wszelkimi sprawami związanymi z informatyką. Stanowi on jak gdyby zarząd, któremu podlegają:

Instytut Maszyn Matematycznych

Instytut Naukowo-Badawczy i Projektowy d.s. Automatyzacji — CNIPIA

Instytut Projektowania Elementów Automatyki i około 10 zakładów budowy elementów i urządzeń automatyki.
IZOT odpowiada ponadto za szkolenie kadr dla potrzeb informatyki.

Ministerstwu Przemysłu Ciężkiego podlegają przemysły: górniczy, metalurgiczny, energetyczny, chemiczny, drzewny. Ministerstwo zajmuje się informatyką w kierowanych przez siebie gałęziach przemysłu. Jeżeli chodzi o przemysł metalurgiczny, to tworzy on koncern hutnictwa żelaza. W skład koncernu wchodzi instytut, w którego gestii scentralizowane są wszelkie prace związane z informatyką w hutnictwie. Koordynuje on między innymi prace w zakresie automatyzacji zbierania informacji, zleca wyższym uczelniom realizację niektórych tematów i pokrywa koszty ich wykonania.

Szereg prac dla hutnictwa realizowany jest przez CNIPIA.

Stan prac w zakresie zastosowania EMC w hutnictwie

Hutnictwo żelaza należy do szybko rozwijających się przemysłów w Bułgarii. W skład jego wchodzą dwie huty:

Kombinat Kremikowski w Kremikowcach obok Sofii

Huta im. Lenina w Pernik.

Produkcja Huty Kremikowskiej wynosi obecnie około 1,5 miliona ton stali. Po rozbudowie, która ma być zakończona w 1975 r., produkcja stali wzrośnie do co najmniej 3 milionów ton. Przewiduje się, że zwolni to Bułgarię od konieczności importu i zezwoli na eksport pewnych ilości wyrobów hutniczych.
W odległości kilku km od Kombinatu położona jest odkrywkowa kopalnia rudy, której zapasy przy założonym wydobyciu rocznym 5 mln ton wystarczą na 50 lat. Obok żelaza ruda zawiera siarczany manganu i ołowiu.

W skład kombinatu wchodzą wydziały:
przygotowania rudy — wzbogacania i spiekania
prażalnie
rozdrabniania
flotacji
aglomerowania
3 koksownie z oddziałami chemicznymi
3 wielkie piece każdy o objętości 1033 m^3
3 piece konwertorowe po 100 t
2 piece elektryczne łukowe po 100 t
walcownia zgniatacz — slabing D 1150
walcownia blachy w kręgach L 1700.

Obecny stan zaawansowania prac nie jest duży, co wynika z krótkiego, bo zaledwie dwuletniego okresu ich prowadzenia. O ile tempo prac nie ulegnie zmianie należy się spodziewać zrealizowania w najbliższych latach planów dotyczących: zbierania danych, planowania operatywnego i przejścia do etapu kierowania produkcją.

W trakcie prowadzenia rozmów strona bułgarska dawała do zrozumienia, że jest zainteresowana w nawiązaniu współpracy ze stroną polską.

Na zakończenie pobytu w Sofii sporządzono odpowiedni protokół oraz plan współpracy na rok 1972. Tematyka planu współpracy nie angażuje żadnej ze stron w dodatkowe prace, a jednocześnie stanowi pomost do nawiązania rzeczywistej współpracy, o ile władze nasze uznają to za słuszne.

Zakupiono licencje na EMC japońskie FALCOM, których produkcję rozpoczęto w ub. roku. Do chwili obecnej nie opanowano produkcji dysków magnetycznych i drukarek wierszowych (przewidują zakup drukarek w Polsce). Uruchomiono serię prototypową EMC jednolitego systemu krajów RWPG R 20. Ponadto otrzymaliśmy nie sprawdzone informacje zakupu 17 szt. EMC firmy IBM (nie znany jest nam stan realizacji).

Rozwój informatyki wiąże się z modernizacją modelu zarządzania. Bułgarzy przystępują do opracowania kompleksowych systemów automatyzacji zarządzania, rejestracji i sterowania procesami produkcyjnymi w najważniejszych dla kraju gałęziach gospodarki.

W Bułgarii istnieje organizacyjne zabezpieczenie projektowania systemów kompleksowych, zakupu sprzętu do realizacji tych systemów oraz dla prac badawczo-wdrożeniowych. W realizacji tych przedsięwzięć bierze udział CNIPIA, który jest instytutem wielozakładowym o znaczeniu ogólnokrajowym i posiada liczną kadrę specjalistów szkoloną w USA, NRF, W. Brytanii, Francji i ZSRR.

Przyznano w obecnym planie pięcioletnim 5 razy więcej środków na automatyzację niż Polska, a 20 razy więcej na jednego mieszkańca.

Wnioski

Na tle poczynionych obserwacji uważamy, że przy obecnym tempie prac — stan informatyki w hutnictwie bułgarskim w ciągu roku—dwu lat będzie wyższy niż w hutnictwie polskim.
Bezwzględnie konieczna jest zmiana w organizacji rozwoju informatyki w polskim hutnictwie. Sytuacja, w której poszczególne huty rozpoczynają i prowadzą cały szereg prac identycznych lub podobnych, każda dla siebie, na swój użytek, a między takimi ośrodka-

mi jak HPMOA w Katowicach, IMŻ w Gliwicach, AGH w Krakowie, OBRI w Warszawie również nie ma współpracy, musi ulec zmianie. O ile HPMOA ma być komórką prowadzącą, musi mieć przede wszystkim pełne poparcie i pomoc władz, opracowany plan długofalowy, zapewnione środki materialne i materiałowe na jego realizację, ukierunkowywać prace poszczególnych komórek.

Konieczna jest rozbudowa bazy technicznej, która dostarczy urządzeń do automatycznego zbierania i przekazywania danych i informacji.
Opieranie wielu systemów o pracę na tradycyjnych dokumentach źródłowych obniża ich efektywność, wiarygodność i zakres przy dużej pracochłonności, a w pewnych wypadkach uniemożliwia ich wdrożenie.

Uważamy za celowe i pożyteczne nawiązanie współpracy ze stroną bułgarską. Współpraca ta mogłaby dotyczyć całokształtu zagadnień związanych z ETO począwszy od opracowania założeń, projektów, dokumentacji przetwarzania i wyposażenia technicznego aż do ostatecznego przekazania systemu po pełnym wdrożeniu do eksploatacji. Strona bułgarska zainteresowana jest dwustronną współpracą w dowolnej formie. W czasie nie zobowiązującej dyskusji byli zainteresowani w:

— utworzeniu bułgarsko-polskiego Instytutu Systemów Automatyzacji (mają 10-piętrowy budynek nie zagospodarowany),

— wysłaniu kilku specjalistów z PRL do wykonania z nimi analizy stanu istniejącego i projektu koncepcyjnego dla Huty Kremikowce,

— kwartalnych seminariów z referowaniem wykonanych prac projektów wdrożeniowych, z wymianą doświadczeń,

— tworzeniu mieszanych zespołów naukowo-badawczych i projektowych w zakresie typowych systemów zastosowania ETO w jednostkach przemysłowych.

Pożądana byłaby również współpraca na odcinku produkowania wyposażenia technicznego.

Rodkiewicz

WĘGRY

W dniach 4—9 października 1971 r. w Węgierskiej Republice Ludowej w miejscowości Veszprem odbyła się III narada specjalistów d.s. zautomatyzowanych systemów zarządzania przedsiębiorstwami przemysłowymi, kombinatami i branżami.

W naradzie uczestniczyli przedstawiciele: Bułgarii, Czechosłowacji, NRD, Polski, Węgier i ZSRR.

Ze strony polskiej udział wzięli: Z. Zapolski, R. Kulesza, T. Łuczyński, I. Chmielewska, D. Kucharski.

Na naradzie omówiono m.in.:

— zagadnienia związane z metodyką projektowania i wdrażania zautomatyzowanych systemów zarządzania (ZSZ)

— zagadnienia intensyfikacji w opracowaniach typowych elementów ZSZ dla Jednolitego Systemu.

Poza programem narady strona węgierska zorganizowała spotkanie z przedstawicielami kierownictwa w Zakładach VIDEOTON w Szekesfeherwar. Na spotkaniu tym zaprezentowano system informatyczny technicznego przygotowania produkcji, planowania produkcji i gospodarki materiałowej stosowany w tych zakładach.

T. Łuczyński

Przegląd prasy zagranicznej

O JAPONII PRAWIE WSZYSTKO (1)

Autorzy tej pracy zbiorowej opatrzonej wyczerpującym tytułem: „Rozwój produkcji i stosowania komputerów w Japonii oraz problematyka programowania w różnych dziedzinach życia gospodarczego i nauki" wyszli z założenia, że prześledzenie japońskiej drogi do dzisiejszego stanu komputeryzacji tego kraju może nam uzmysłowić, jak należy to robić. Japonia wysunęła się ostatnio na drugie miejsce w świecie pod względem liczby użytkowanych komputerów. Ale jej początki w tej dziedzinie również były trudne... Jednak w chwili obecnej Japonia wyprzedziła Stany Zjednoczone w stosowaniu komputerów do sterowania procesami produkcyjnymi; USA bowiem w dalszym ciągu preferują obsługę handlu i finansów.

Ponieważ liczące ponad 70 stron opracowanie ukazało się dotychczas w niewielkiej liczbie egzemplarzy powielonego maszynopisu, postanowiliśmy w kolejnych numerach „INFORMATYKI" przedstawiać Czytelnikom wybrane zagadnienia informatyki japońskiej, oczywiście w oparciu o wyżej wymienioną pracę zbiorową.

Charakterystyka ogólna japońskiego przemysłu komputerów

Zaczęło się w 1950 roku. Pierwszy japoński model komputera pochodzący już z produkcji przemysłowej ukazał się w 1958 r., to jest w 7 lat po UNIVAC-1 i IBM. Japończycy natychmiast zrozumieli, że z każdym następnym rokiem ich dystans do czołówki światowej będzie się zwiększał wielokrotnie. Postawili więc z miejsca na rozwój produkcji komputerów w oparciu o najlepsze technologie zagraniczne. Podpisaniu odpowiednich umów, szczęśliwie dla Japonii, towarzyszyło wygaśnięcie podstawowego patentu firmy IBM. Był to rok 1960, a już w cztery lata później znalazła się Japonia na liście najbardziej zaawansowanych w tej dziedzinie krajów.

Za licencjami i nakładami na badania własne poszło niezmiernie, jak się później okazało, fortunne pociągnięcie w wewnętrznej organizacji eksploatowania maszyn cyfrowych. Otóż powołano do życia przedsiębiorstwo pod nazwą „Japan Electronic Computer Company" (IECC), które do dziś zajmuje się dzierżawieniem komputerów poszczególnym firmom produkcyjnym. System dzierżawczy — r e n t a l s y s t e m — zmniejsza obciążenia finansowe u producentów oraz — poprzez konsolidację systemu — ułatwia rozprowadzanie komputerów wśród użytkowników.

Trzecim posunięciem była i jest nadal rządowa opieka nad japońskim przemysłem komputerów oraz ochrona rynku krajowego. Cała ta wstępna działalność na rzecz komputeryzacji kraju wypływała ze świadomości, że komputery są podstawą rozwoju informatyki, a ich produkcja jako wiodąca w przemyśle elektronicznym ma wielki wpływ na cały przemysł, choćby w postaci źródła systemów technologicznych.

Drugie, po Stanach Zjednoczonych, miejsce w produkcji komputerów zapewniają Japonii następujące firmy:

- FUJITSU LIMITED, COMMUNICATIONS AND ELECTRONICS
- MITSUBISHI ELECTRIC CORPORATION
- HITACHI LTD
- NIPPON ELECTRIC COMPANY Ltd
- OKI ELECTRIC INDUSTRY COMPANY Ltd
- TOKYO SHIBAURA ELECTRIC COMPANY Ltd
- IBM — Japan

Wszystkie, prócz IBM — Japan, towarzystwa to producenci japońscy o kapitale japońskim. Niezależnie od nich istnieją firmy specjalizujące się wyłącznie w produkcji „komputerów kontrolnych". Są to:

- HOKUSHIN ELECTRIC WORKS Ltd
- MITSUBISHI PRECISION COMPANY Ltd
- SHIMAZU SEISAKUSHO Ltd
- YASUKAWA ELECTRIC Mfg. Co., Ltd
- YOKOGAWA ELECTRIC WORKS Ltd

Ponadto następujące firmy zajmują się przemysłem niewielkich maszyn typu biurowego:

- MATSUSHITA COMMUNICATION INDUSTRIAL Co., Ltd
- THE JAPAN RADIO Co., Ltd
- USAC DENSHI KOGYO

Urządzenia peryferyjne wytwarzają: Teshiba Ampex Co. (urządzenia taśm magnetycznych) i Hitachi Koki Co., Ltd (linie drukarskie). Wymienić także należy wśród producentów samych komputerów firmę OKI-UNIVAC, która jednak nie jest w pełni samodzielna, gdyż większą część swej produkcji sprzedaje towarzystwu Oki Electric Industry Co. Ltd.

O wielkości produkcji komputerów w Japonii oraz — szerzej — o interesującym systemie dzierżawczym i sprzedaży sprzętu na rynku krajowym — w następnym numerze.

Oprac. ELK

Brytyjskie Towarzystwo Komputerowe

Informatycy za granicą

BCS — British Computer Society [1]) — powstało w 1957 r. jako stowarzyszenie naukowo-techniczne grupujące praktyków i teoretyków „wiedzy komputerowej", zwanej dziś już często informatyką. Przez pierwsze 10 lat swego istnienia BCS zdobyło blisko 20 tysięcy członków i uzyskało międzynarodowe uznanie jako wydawca kwartalnika Computer Journal — zaliczanego do pięciu najważniejszych w świecie czasopism komputerowych. W odróżnieniu od wielu typowo brytyjskich stowarzyszeń o charakterze zawodowym — pielęgnujących wielostopniowe systemy członkostwa i broniących dopływu laików wysokimi wymaganiami kwalifikacyjnymi — BCS przyjmowało wówczas na członka każdą osobę deklarującą zainteresowanie rozwojem komputerów i ich wpływem na współczesną cywilizację.

W jedenastym roku swego istnienia Towarzystwo postanowiło radykalnie zmniejszyć udział osób ubocznie tylko zainteresowanych problemami komputeryzacji — w dodatku na liście członków figurujących często już tylko jako „martwe dusze", zalegające z opłacaniem składek rocznych. Wprowadzono wówczas egzaminy kwalifikacyjne, dosyć obszerne, chociaż zróżnicowane według zakresu zainteresowań egzaminowanego. W ten sposób zlikwidowano „deficyt wydawniczy" — członkowie bowiem wszystkie oficjalne publikacje Towarzystwa otrzymują za darmo — ale jednocześnie zmniejszyła się w katastrofalny sposób suma wpływów ze składek członkowskich. Ostatecznie więc, z początkiem bieżącego roku, wprowadzono szereg dodatkowych klauzul statutowych, które umożliwiają uzyskanie najwyższych stopni członkostwa po wylegitymowaniu się odpowiednim stopniem naukowym, pozycją kierowniczą oraz wieloletnim stażem. Jednocześnie zarząd Towarzystwa apelował do członków, zwłaszcza niższych stopni, aby w przypadku tylko marginesowego zainteresowania sprawami komputeryzacji rezygnowali dobrowolnie z bezpłatnej prenumeraty wydawnictw. Ostatecznie, zresztą po szeregu burzliwych dyskusji wewnętrznych na zebraniach sprawozdawczo-wyborczych, wyprowadzono gospodarkę Towarzystwa na spokojne wody i obecnie lista członków obejmuje ponad 16 tys. członków, zgrupowanych w 32 oddziałach krajowych i 2 zagranicznych (Hong-Kong oraz Paryż).

Siedmiostopniowe członkostwo

I. Najwcześniej można przystąpić do Towarzystwa po ukończeniu lat 17, o ile obrany kierunek studiów wyższych lub pomaturalnych (college) wskazuje na zainteresowanie komputerowe petenta, który wówczas otrzymuje kategorię studenta i płaci najniższą składkę (3 funtów rocznie). Członkiem-studentem można być najdłużej 4 lata, po którym to okresie należy się poddać niższemu egzaminowi kwalifikacyjnemu.

II. Ukończenie 21 lat i zdanie niższego egzaminu — przy braku 5 lat praktyki komputerowej — uprawnia do statusu kandydata, płacącego już składkę 5-funtową.

III. Następnym stopniem przejściowym jest członek-stowarzyszony, który na tym statusie może znaleźć się po ukończeniu 25 lat, zaś pozostawać tak długo, póki nie nabierze wyższych uprawnień (płacąc przez cały czas już składkę 7-funtową), wiążących się z odpowiednimi egzaminami.

IV. Bez żadnego egzaminu można być członkiem-delegatem z ramienia organizacji handlowej, jednostki przemysłowej, instytutu, stowarzyszenia lub innej instytucji. Obowiązuje tutaj zakaz częstszej wymiany delegatów niż raz na rok. Składka roczna za dwu delegatów wynosi £ 20. Podczas głosowania obaj mają prawo do „pół-głosów".

V. Po ukończeniu 25 lat, mając za sobą zdany egzamin kwalifikacyjny oraz mogą się wylegitymować 5-letnim stażem komputerowym — a także poparciem jednego członka-seniora oraz jednego członka zwyczajnego — uzyskuje się status członka-juniora (LBCS). Składka wtedy wynosi już 8 £.

VI. Jeżeli jednak egzaminy wypadły szczególnie dobrze, zaś petent ma 5 lat stażu jako programista i analityk (lub 7 lat stażu komputerowego i wykształcenie ukierunko-

Tablica

KRYTERIA KWALIFIKOWANIA NOWYCH CZŁONKÓW BCS. SCHEMAT UPROSZCZONY

Wyszczególnienie	Reguły kwalifikacyjne (po 25.03.68)									
	a	b1	b2	c	d	e1	e2	f1	f2	g
Warunki indywidualne kandydata										
1. Minimum wieku	17	21	25	21	25	25		30		21
2. Maksimum wieku	21	24	.	.	.	.		.		.
3. Zaliczenie egzaminów (Part I a. II)	—	I		—	I	I+II		II+I		..
4. Odbywanie studiów wyższych	+	—		.	.	dyplom	—	.		.
5. Staż komputerowy	—	.	5	.	4	4	6	7	.	.
6. Staż kierowniczy	—	.		.	.	.		4	.	.
7. Nominacja pracodawcy	—	—		—	—	—		—		+
8. Poparcie 2 członków tytularnych	—	MM		—	MF	MF		FF	.	—
9. Członkostwo Komitetu Założycielskiego	—	—		—	—	—		—	+	—
Kategoria członkostwa i składka										
I. Student	3	—		—	—	—		—		—
II. Associate	—	5		—	—	—		—		—
III. Affiliate	—	—		7	—	—		—		—
IV. Licentiate	—	—		—	8	—		—		—
V. Member	—	—		—	—	9		—		—
VI. Fellow	—	—		—	—	—		12		—
VII. Institutional Delegate	—	—		—	—	—		—		20

Oznaczenia symboliczne:
+ Tak
— Nie
. Pominięte
I Niższy egzamin BCS
II Wyższy egzamin BCS

[1]) Obecna siedziba Zarządu Głównego: 29 Portland Place, London W1N 4AP, Anglia.

wane, względnie jakąkolwiek 10-letnią praktykę komputerową nawet bez wykształcenia) — przy analogicznym poparciu, jak uprzednio, może ubiegać się o status M.B.C.S. — członka zwyczajnego (£ 9).

VII. Ten zaś, kto legitymuje się 5-letnim stażem kierowniczym oraz 8-letnim stażem komputerowym — i był członkiem zwyczajnym a także uzyskał poparcie dwu członków-seniorów (FBCS) — może ubiegać się o status takiegoż. Tutaj limit wieku wynosi aż 30 lat. Składka roczna — £ 12.

Cele statutowe

Jako naczelne zadanie Brytyjskie Towarzystwo Komputerowe postuluje:

— formułowanie i kultywowanie wysokich standardów kompetencji zawodowej,

— rozpowszechnianie wiedzy komputerowej we wszelkich jej aspektach hardwerowych (cyfrowych i analogowych) jak i softwerowych (obliczeniowych, sterowaniowych i zarządzeniowych),

— ułatwianie nie skrępowanej wymiany informacji i poglądów środowiska komputerowego w sposób maksymalnie „kulturalny" w sensie zawodowym,

— prowadzenie egzaminów kwalifikacyjnych z zakresu wiedzy komputerowej.

Poszczególnych członków Towarzystwa, stosownie do powyższego, obowiązuje:

— przyjmowanie pełnej odpowiedzialności za wykonywane prace, tj. działając przez zaprzeczenie, niepodejmowanie się prac, za które nie mogą przyjąć pełnej odpowiedzialności,

— zachowywanie się zgodnie z etyką zawodową zgodnie z najlepszymi tradycjami brytyjskimi w tym zakresie,

— poszanowanie tak własnej jak i cudzej tajemnicy zawodowej,

— całkowita bezstronność w przypadku świadczenia doradztwa.

Powyższe cele Towarzystwo osiąga przez:

— doradztwo na rzecz instytucji rządowych i innych ciał społecznych w interesie ogólnonarodowym,

— organizowanie konferencji i spotkań zawodowych,

— popieranie grup roboczych do określonych zagadnień,

— organizowanie terenowego życia zawodowego,

— współdziałanie z instytucjami krajowymi i zagranicznymi o zbliżonych zainteresowaniach.

Obecnie Towarzystwo wydaje dwie publikacje regularne: kwartalnik COMPUTER JOURNAL oraz miesięcznik COMPUTER BULLETIN. Posiada w każdej jednostce administracji terenowej i większych aglomeracjach przemysłowych swoje oddziały, a nawet dwa zagraniczne (jeden w Hong-Kongu i drugi w Paryżu). Ponadto Towarzystwo popiera finansowo działalność blisko 40 różnotematycznych grup roboczych do zagadnień, m. in.:

— wyższych języków programowania
— obliczeń analogowych
— sztucznej inteligencji
— dydaktyki komputerowej
— rehabilitacji ociemniałych
— systemów MIS
— projektowania komputerowego
— sztuki komputerowej
— prawa obliczeniowego
— teledacji
— wyszukiwania informacji
— urządzeń wejścia/wyjścia
— projektowania procesorów
— optymalizacji liniowej
— diagnostyki medycznej
— systemów abonenckich
— kontroli jakości w produkcji
— planowania przestrzennego
— skutków społecznych komputeryzacji
— zalecanych metod przetwarzania itd.

Działalność grup znajduje odbicie w organizowanych co tydzień lub rzadziej spotkaniach zawodowych w różnych miastach. Ponadto grupy przygotowują referaty specjalistyczne na organizowanych co dwa lata konferencjach DATAFAIR.

A. B. Empacher (MBCS)

Informatyka w bułgarskim przemyśle maszynowym

Władze bułgarskie wytypowały w 1969 r. region Warny jako teren eksperymentalnego wdrażania zautomatyzowanych systemów zarządzania. Tam również rozpoczną się prace nad jednolitym systemem przetwarzania informacji ekonomicznych, wedle zatwierdzonego projektu. Informatyka w pierwszym rzędzie wkroczy do nowo budowanych lub modernizowanych przedsiębiorstw, w których organizacja pracy odpowiada nowoczesnym wymogom. W innych przedsiębiorstwach stopniowo będzie się wdrażać systemy zautomatyzowanego przetwarzania danych, po okresie wstępnego przygotowania. Projekty systemów opracowuje centralny instytut ORGPROJEKT.

Dla fabryki manometrów MORZE CZARNE przewiduje się wykończenie projektu roboczego i wdrożenie zautomatyzowanego systemu przetwarzania danych w 1973 r. W jego skład wchodzą następujące podsystemy: sterowanie technicznym przygotowaniem produkcji, średnioterminowe planowanie produkcji i sprawozdawczość, operatywne planowanie produkcji i dyspozycja, gospodarka maszynowa, gospodarka materiałowa, praca i płace, działalność finansowa. System ma na celu optymalizację programu produkcyjnego i planowanie techniczne, produkcyjne i finansowe z pomocą komputera na podstawie jednolitej bazy normatywnej. Podkreśla się niektóre oryginalne rozwiązania takie jak: metoda specyfikacji zagregowanej, kontrola zapasów magazynowych ze współczynnikiem zabezpieczenia materiałowego produkcji, system preferencji itd. Oprócz niewymiernych korzyści oczekuje się efektów ekonomicznych w wysokości 60 tys. lewów rocznie. Zwrot nakładów ma nastąpić w ciągu dwóch lat.

Inny zakład produkcyjny przemysłu maszynowego METAL rozbudowuje się obecnie i modernizuje z pomocą dostaw urządzeń i dokumentacji z Włoch. Przewiduje się, że do 1979 r. nastąpi czterokrotny wzrost wartości produkcji. Zautomatyzowany system zarządzania będzie wdrażany w dwóch etapach. W pierwszym okresie będą z pomocą firmy włoskiej wdrożone niektóre podsystemy takie jak: techniczne przygotowanie produkcji, praca i płaca, planowanie, finansowanie i rachunek kosztów, sterowanie produkcją itd. W drugim etapie ORGPROJEKT zamierza wprowadzić tam podsystemy gospodarki narzędziowej, remontowej, kontroli jakościowej oraz podsystem działalności administracyjno-gospodarczej. Zakład ma już zainstalowany komputer IBM 360 model 20, a w 1973 r. ma otrzymać większą maszynę z serii RIAD. Powstanie tam ośrodek obliczeniowy obsługujący różne przedsiębiorstwa w regionie Warny.

W zakładach WULKAN i ELPROM przygotowuje się reorganizację zarządzania, a jednocześnie opracowuje się zautomatyzowane systemy, których projekty robocze miały być zakończone do końca 1971 r. Zakłady te będą korzystały z obcych ośrodków obliczeniowych.

„Mechanizacija i Awtomatizacija Proizwodstwa", 1971, nr 10.

D. P.

PRZEGLĄD WYDAWNICTW

Infrainformator ISIS[1]

Strukturotomia — jak można by próbować przetłumaczyć idiomatyczne określenie unbundling, odnoszące się do coraz częściej praktykowanego rozdziału kosztu oprogramowania użytkowego od kosztu „gołego" komputera z oprogramowaniem podstawowym — spowodowała niewątpliwy wzrost zainteresowania programami i systemami jako towarem. Niejako odpowiedzią na tego rodzaju „zapotrzebowanie społeczne" jest nowo powstałe wydawnictwo ISIS — Infratest Software Informations Service, wydawane przez firmę INFRATEST w Monachium.

Niewątpliwie ISIS jest informatorem handlowym — a więc ma za zadanie orientować swych czytelników w warunkach zakupu lub wynajmu poszczególnych programów i ich pakietów systemowych. Tak więc okaże się nikłą pomocą dla „ambitnego" projektanta, który zechce „sam", od początku do końca, być twórcą swego systemu. Informator ISIS bowiem tylko w kilku przypadkach podaje schematy blokowe, a w dodatku znacznie uproszczone, których adaptacja jest iluzoryczna. Natomiast projektant „bez ambicji", który w dodatku dysponowałby odpowiednimi środkami dewizowymi — będzie traktował ISIS jako cenne źródło naprowadzające, szczegóły zaś omówi bezpośrednio z właścicielem interesującego go programu czy też systemu.

Całość Infrainformatora składa się z dwu zasadniczych części — firmowej (F) i programowej (P). W części F wyodrębniono działy:

1 — firmy prowadzące działalność doradczą w dziedzinie programowania (adresy, zakres przedmiotowy, usługi komputerowe, liczba zatrudnionych, cennik konsultacji i ekspertyz, łącznie blisko 100 firm z terenu NRF, Austrii i Szwajcarii);

2 — firmy wydzierżawiające sprzęt komputerowy (adresy, typy maszyn, języki programowania, biblioteki programów, cenniki);

3 — firmy organizujące kursy programowania i projektowania (adresy, terminarz kursów, cenniki).
Część P obejmuje trzy dalsze działy:

4 — programy z zakresu przetwarzania danych (księgowość finansowa, gospodarka materiałowa, produkcja wyrobów gotowych, marketing, banki danych, wyszukiwanie informacji, gry dyrektorskie itd.);

5 — programy z zakresu nauki i techniki (matematyka stosowana, badania operacyjne, mechanika, geodezja, optyka, akustyka, budownictwo, elektrotechnika, projektowanie graficzne i in.);

6 — systemy programowania (generatory, testery, systemy eksploatacyjne, podprogramy podstawowe).

W poddziałach powyższej klasyfikacji daje się jeszcze zauważyć roboczą prowizorkę — być może jednak, że zostanie ona usunięta w miarę zwiększenia się objętości wydawnictwa. Uzupełnieniem informatora są indeksy — firmowy i programowy (dział 1) — oraz aneksy (dział 8), obejmujące m.in. przykład wypełniania standardowego formularza ISIS (zgłoszenia programu do informatora), szczegółowe warunki współpracy z firmą INFRATEST, warunki prenumeraty[2]) oraz kartki korespondencyjne.

Szczegółów dotyczących rozliczeń finansowych między INFRATEST a reklamowanymi w informatorze firmami — nie podano.

ABE

---

[1]) ISIS—1971, Infratest Software Information Service, Informationen über Software-Produkte für die Bundesrepublik Deutschland, Österreich und Schweitz; wydawca INFRATEST Informations-Forschung 8000 München 21, Lendsbergerstr. 338; prenumerata roczna 375 DM (3 zeszyty), pojedynczy numer 150 DM.

[2]) O ile wiadomo, prenumeratę informatora ISIS na rok 1972 zgłosiła biblioteka Centrum ETOB w Warszawie, ul. Ogrodowa 28, tel. 20-48-21 wewn. 210.

Bibliografia książek polskich z dziedziny informatyki

● **Elektroniczna maszyna cyfrowa ODRA 1304. Dokumentacja techniczno-ruchowa. Urządzenia. Perforator taśmy papierowej PT 304-1.** Wydawnictwo Wrocławskich Zakładów Elektronicznych ELWRO. Wrocław 1970, s. 21.

Opis techniczny: opis ogólny, opis działania układów elektronicznych perforatora, pulpity, zasilanie, zasady obsługi perforatora. Instrukcja demontażu i pakowania oraz rozpakowania i montażu. Schematy logiczne. Rysunki pakietów. Schematy montażowe. Rysunki powołane. Podręcznik jest przeznaczony dla operatorów i konserwatorów maszyny cyfrowej ODRA 1304.

● **Elektroniczna maszyna cyfrowa ODRA 1304. Dokumentacja techniczno-ruchowa. Oprogramowanie. PLAN. Podręcznik programowania. T. 1. PLAN — podstawowy język programowania.** Wyd. Wrocławskich Zakładów Elektronicznych ELWRO, Wrocław 1970, wlb.

Cz. 1. Informacje podstawowe: informacje podstawowe dotyczące maszyn serii 1300 (słowa i podziały słów, nadmiar i przeniesienie, tryby, modyfikacja, modyfikatory ujemne, wieloprogramowość itd.), podstawowe informacje o języku PLAN (ogólna postać programu, wykorzystanie symboli w języku PLAN, podział pamięci operacyjnej, arkusz programowy języka PLAN, postacie zapisu danych, ogólna postać zapisu instrukcji, postacie operandów w zapisach instrukcji programowych, zestawienie ułatwień programowania), dodatkowe ułatwienia programowania (wprowadzenie, subprogramowanie, segmentacja programów). Cz. 2. Instrukcje i dyrektywy: instrukcje, pseudoinstrukcje, dyrektywy.

Podręcznik jest przeznaczony dla programistów maszyny ODRA 1304, którzy ukończyli kurs programowania w języku PLAN.

● **Elektroniczna maszyna cyfrowa ODRA 1304. Dokumentacja techniczno-ruchowa. Oprogramowanie. Podstawowe urządzenia zewnętrzne. Programowanie.** Wydawnictwo Wrocławskich Zakładów Elektronicznych ELWRO, Wrocław 1971, s. 91.

Wstęp. Informacje wprowadzające. Cz. 1. Programowanie na poziomie executive: sterowanie podstawowymi urządzeniami zewnętrznymi, przesyłanie danych, czytanie kart, czytanie taśmy papierowej, dziurkowanie taśmy papierowej, drukowanie na drukarce wierszowej.
Cz. 2. Oprogramowanie: generator wejścia/wyjścia, system podprogramów wejścia/wyjścia. Załączniki: system podprogramów wejścia/wyjścia, kody zewnętrzne i wewnętrzne. Podręcznik przeznaczony dla programistów maszyny ODRA 1304. Przy korzystaniu z niego wymagana jest znajomość programowania w języku PLAN.

cd. na IV skrz.